

I codici: teoria dell'informazione e codifica

Indice Generale

Fondamenti Matematici della Teoria dei Codici

Codici a Blocco Lineari

Codici Ciclici e BCH

Codici Convoluzionali

Codici LDPC e Turbo

Teoria dell'Informazione di Shannon

Codici Crittografici

Applicazioni Moderne

1. FONDAMENTI MATEMATICI DELLA TEORIA DEI CODICI

1.1 Spazi Vettoriali su Campi Finiti

1.1.1 Campi di Galois $GF(q)$

Un campo finito o campo di Galois $GF(q)$ è un insieme finito di q elementi dotato di due operazioni (addizione e moltiplicazione) che soddisfano gli assiomi di campo.

Teorema 1.1 (Esistenza dei Campi Finiti):

Un campo finito $GF(q)$ esiste se e solo se $q = p^m$, dove p è un numero primo e $m \geq 1$ è un intero.

Dimostrazione:

La caratteristica di un campo finito deve essere un numero primo p . Ogni campo finito contiene un sottocampo isomorfo a $\mathbb{Z}_p$. Se $|GF(q)| = q$, allora $GF(q)$ è uno spazio vettoriale su $\mathbb{Z}_p$ di dimensione m , quindi $q = p^m$. $\square$

Costruzione di $GF(2^m)$:

Sia $f(x)$ un polinomio irriducibile di grado m su $GF(2)$:

$$f(x) = x^m + c_{m-1}x^{m-1} + \dots + c_1x + c_0$$

dove $c_i \in \{0,1\}$.

Allora $GF(2^m) = GF(2)[x] / (f(x))$

Gli elementi sono rappresentati come:

$$a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \dots + a_1x + a_0, \text{ dove } a_i \in GF(2)$$

Esempio 1.1: Costruzione di $GF(2^3)$

Il polinomio $f(x) = x^3 + x + 1$ è irriducibile su $GF(2)$.

Gli 8 elementi di $GF(2^3)$ sono:

$$0 = 0$$

$$1 = 1$$

$$\alpha = x$$

$$\alpha^2 = x^2$$

$$\alpha^3 = x + 1 \text{ (poiché } x^3 \equiv x + 1 \pmod{f(x)})$$

$$\alpha^4 = x^2 + x$$

$$\alpha^5 = x^2 + x + 1$$

$$\alpha^6 = x^2 + 1$$

Tavola di moltiplicazione:

$$\alpha^7 = \alpha \cdot \alpha^6 = \alpha \cdot (x^2 + 1) = x^3 + x = (x+1) + x = 1$$

Quindi α è un elemento primitivo di ordine 7.

1.1.2 Polinomi su Campi Finiti

Definizione 1.2: Un polinomio $p(x)$ di grado m su $GF(q)$ è irriducibile se non può essere fattorizzato come prodotto di polinomi di grado inferiore.

Teorema 1.2 (Numero di Polinomi Irriducibili):

Il numero $N_q(m)$ di polinomi monici irriducibili di grado m su $GF(q)$ è:

$$N_q(m) = (1/m) \cdot \sum_{(d|m)} \mu(m/d) \cdot q^d$$

dove μ è la funzione di Möbius:

$\mu(n) = 1$ se n è prodotto di un numero pari di primi distinti

$\mu(n) = -1$ se n è prodotto di un numero dispari di primi distinti

$\mu(n) = 0$ se n ha fattori quadrati

Esempio 1.2: Polinomi irriducibili su $GF(2)$ di grado 4:

$$N_2(4) = (1/4)[\mu(4) \cdot 2^1 + \mu(2) \cdot 2^2 + \mu(1) \cdot 2^4]$$

$$= (1/4)[0 \cdot 2 + (-1) \cdot 4 + 1 \cdot 16]$$

$$= (1/4)[0 - 4 + 16]$$

$$= 3$$

I tre polinomi sono:

$$x^4 + x + 1$$

$$x^4 + x^3 + 1$$

$$x^4 + x^3 + x^2 + x + 1$$

1.2 Metriche e Distanze

1.2.1 Distanza di Hamming

Definizione 1.3: La distanza di Hamming $d_H(x,y)$ tra due vettori $x,y \in GF(q)^n$ è il numero di coordinate in cui differiscono:

$$d_H(x,y) = |\{i : x_i \neq y_i, 1 \leq i \leq n\}|$$

Proprietà:

$d_H(x,y) \geq 0$, con uguaglianza se e solo se $x = y$

$d_H(x,y) = d_H(y,x)$ (simmetria)

$d_H(x,z) \leq d_H(x,y) + d_H(y,z)$ (disuguaglianza triangolare)

Peso di Hamming:

$w_H(x) = d_H(x, 0)$ = numero di componenti non nulle di x

Proposizione 1.1: Per codici lineari su $GF(q)$:

$$d_H(x, y) = w_H(x - y)$$

1.2.2 Distanza Minima di un Codice

Definizione 1.4: La distanza minima d di un codice C è:

$$d = \min\{d_H(x, y) : x, y \in C, x \neq y\}$$

Per codici lineari:

$$d = \min\{w_H(c) : c \in C, c \neq 0\}$$

Teorema 1.3 (Capacità di Correzione):

Un codice con distanza minima d può:

Rilevare fino a $d-1$ errori

Correggere fino a $t = \lfloor (d-1)/2 \rfloor$ errori

Dimostrazione:

Se c è la parola codificata trasmessa e r è quella ricevuta con $e \leq t$ errori, allora:

$$d_H(r, c) = e \leq t < d/2$$

Per ogni altra parola codificata $c' \neq c$:

$$d_H(r, c') \geq d_H(c, c') - d_H(r, c) \geq d - t > d/2 > d_H(r, c)$$

Quindi c è l'unica parola codificata più vicina a r . $\square$

1.3 Limiti Fondamentali

1.3.1 Limite della Sfera

Teorema 1.4 (Hamming Bound o Sphere-Packing Bound):

Un codice q -ario (n, M, d) con M parole di lunghezza n e distanza minima d soddisfa:

$$M \cdot \sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i \leq q^n$$

dove $t = \lfloor (d-1)/2 \rfloor$.

Dimostrazione:

Le sfere di raggio t centrate in parole codificate distinte sono disgiunte. Ogni sfera contiene:

$$V_q(n, t) = \sum_{i=0}^t \binom{n}{i} \cdot (q-1)^i$$

vettori. Esistendo M sfere disgiunte in uno spazio di q^n vettori totali, si ha $M \cdot V_q(n, t) \leq q^n$. $\square$

Definizione 1.5: Un codice che raggiunge il limite di Hamming è detto codice perfetto.

Esempi di codici perfetti:

Codici di ripetizione: $(n, 2, n)$ su $GF(2)$ con n dispari

Codici di Hamming: $(2^m - 1, 2^{2^m - 1 - m}, 3)$

Codice di Golay binario: $(23, 2^{12}, 7)$

Codice di Golay ternario: $(11, 3^6, 5)$

1.3.2 Limite di Singleton

Teorema 1.5 (Singleton Bound):

Per ogni codice (n, M, d) su un alfabeto di cardinalità q :

$$M \leq q^{(n-d+1)}$$

Equivalentemente, per un codice lineare $[n, k, d]$:

$$k \leq n - d + 1$$

oppure:

$$d \leq n - k + 1$$

Dimostrazione:

Proiettando le M parole codificate sulle prime $n-d+1$ coordinate, esse devono essere tutte distinte (altrimenti due parole concorderebbero sulle prime $n-d+1$ posizioni e avrebbero distanza $\leq d-1$). Esistono $q^{(n-d+1)}$ possibili proiezioni. $\square$

Definizione 1.6: Un codice che raggiunge il limite di Singleton è detto MDS (Maximum Distance Separable).

Per codici MDS: $d = n - k + 1$

Esempio 1.3: Codici Reed-Solomon sono MDS.

1.3.3 Limite di Plotkin

Teorema 1.6 (Plotkin Bound):

Per un codice binario (n, M, d) con $2d > n$:

$$M \leq 2d / (2d - n)$$

Se inoltre d è pari e $2d = n$, allora:

$$M \leq 2(d+1)$$

Dimostrazione:

Sia S la somma di tutte le distanze a coppie:

$$S = \sum_{(1 \leq i < j \leq M)} d_H(c_i, c_j)$$

Contando per posizioni: in ogni posizione, se ci sono x zeri e $M-x$ uni, il contributo alla somma totale è $x(M-x)$. Massimizzando:

$$S \leq n \cdot (M^2/4)$$

D'altra parte, ogni coppia contribuisce almeno d :

$$S \geq \binom{M}{2} \cdot d = M(M-1)d/2$$

Quindi:

$$M(M-1)d/2 \leq nM^2/4$$

$$M-1 \leq n \cdot M / (2d)$$

$$M(2d-n) \leq 2d$$

$$M \leq 2d / (2d-n) \quad \square$$

1.3.4 Limite di Gilbert-Varshamov

Teorema 1.7 (Gilbert-Varshamov Bound):

Esiste un codice q -ario (n, M, d) tale che:

$$M \geq q^n / V_q(n, d-1)$$

dove $V_q(n, r) = \sum_{i=0}^r \binom{n}{i} (q-1)^i$.

Per codici lineari $[n, k, d]$ su $GF(q)$:

$$q^k \geq q^n / V_q(n, d-1)$$

Dimostrazione (costruttiva tramite algoritmo greedy):

Costruiamo il codice incrementalmente:

Iniziamo con $C = \{0\}$

Finché esistono parole $x \in GF(q)^n$ tali che $d_H(x, c) \geq d$ per ogni $c \in C$, aggiungiamo x a C

Al termine, ogni parola non in C è a distanza $< d$ da qualche parola in C , quindi le sfere di raggio $d-1$ centrate nelle parole di C coprono tutto lo spazio:

$$|C| \cdot V_q(n, d-1) \geq q^n$$

Quindi $|C| \geq q^n / V_q(n, d-1)$. $\square$

2. CODICI A BLOCCO LINEARI

2.1 Definizioni e Proprietà Fondamentali

2.1.1 Codici Lineari

Definizione 2.1: Un codice lineare $[n, k, d]$ su $GF(q)$ è un sottospazio k -dimensionale di $GF(q)^n$ con distanza minima d .

Notazione: $C \subseteq GF(q)^n$, $\dim(C) = k$

Proprietà fondamentali:

$0 \in C$ (elemento neutro)

Se $c_1, c_2 \in C$ allora $c_1 + c_2 \in C$ (chiusura additiva)

Se $c \in C$ e $\alpha \in GF(q)$ allora $\alpha \cdot c \in C$ (chiusura scalare)

Tasso del codice:

$$R = k/n$$

rappresenta la frazione di simboli informativi.

Ridondanza:

$$\rho = (n-k)/n = 1 - R$$

2.1.2 Matrice Generatrice

Definizione 2.2: Una matrice generatrice G di un codice $[n, k]$ è una matrice $k \times n$ di rango k le cui righe formano una base per C :

$$G = [g_1, g_2, \dots, g_k]^T \in GF(q)^{(k \times n)}$$

La codifica è data da:

$$c = m \cdot G$$

dove $m \in GF(q)^k$ è il messaggio e $c \in C$ è la parola codificata.

Forma sistemática:

$$G_{\text{sys}} = [I_k \mid P]$$

dove I_k è la matrice identità $k \times k$ e P è una matrice $k \times (n-k)$.

In forma sistemática:

$$c = [m \mid m \cdot P] = [\text{informazione} \mid \text{ridondanza}]$$

Proposizione 2.1: Ogni codice lineare ha una matrice generatrice in forma sistemática (a meno di permutazioni di colonne).

Dimostrazione:

Mediante eliminazione gaussiana, possiamo ridurre G in forma a scala. Permutando le colonne, possiamo ottenere k colonne linearmente indipendenti nelle prime k posizioni, formando I_k . $\square$

2.1.3 Matrice di Controllo di Parità

Definizione 2.3: Una matrice di controllo di parità H di un codice $[n, k]$ è una matrice $(n-k) \times n$ di rango $n-k$ tale che:

$$C = \{c \in GF(q)^n : H \cdot c^T = 0\}$$

Teorema 2.1 (Relazione G-H):

Se $G = [I_k \mid P]$ è in forma sistemática, allora:

$$H = [-P^T \mid I_{(n-k)}]$$

è una matrice di controllo di parità.

Dimostrazione:

$$\begin{aligned} H \cdot G^T &= [-P^T \mid I_{(n-k)}] \cdot [I_k; P^T] \\ &= -P^T + P^T \\ &= 0 \quad \square \end{aligned}$$

Teorema 2.2 (Distanza Minima e Matrice H):

La distanza minima d di un codice lineare soddisfa:

$$d = \min\{j : \text{esistono } j \text{ colonne linearmente dipendenti di } H\}$$

Equivalentemente:

$$d \geq \delta \iff \text{ogni } \delta-1 \text{ colonne di } H \text{ sono linearmente indipendenti}$$

Dimostrazione:

Una parola codificata c ha peso w se e solo se ci sono esattamente w posizioni $i_1, \dots, i_w$ con $c_{(i_j)} \neq 0$.

$H \cdot c^T = 0$ implica:

$$\sum_{j=1}^w c_{(i_j)} \cdot h_{(i_j)} = 0$$

dove h_i è la i -esima colonna di H . Questo è possibile se e solo se le colonne $h_{(i_1)}, \dots, h_{(i_w)}$ sono linearmente dipendenti. $\square$

2.2 Codici di Hamming

2.2.1 Costruzione e Parametri

Definizione 2.4: Un codice di Hamming $\text{Ham}(r,q)$ è un codice lineare su $\text{GF}(q)$ con matrice di controllo di parità le cui colonne sono tutti i vettori non nulli di $\text{GF}(q)^r$, uno per ogni classe di equivalenza modulo moltiplicazione scalare.

Parametri: $[n,k,d] = [(q^r-1)/(q-1), (q^r-1)/(q-1)-r, 3]$

Per $q=2$ (codici binari):

$$n = 2^r - 1$$

$$k = 2^r - r - 1$$

$$d = 3$$

Esempio 2.1: Codice di Hamming $\text{Ham}(3,2) = [7,4,3]$

Matrice di controllo di parità:

$$H = [0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1]$$

$$[0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1]$$

$$[1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1]$$

Le colonne rappresentano i numeri da 1 a 7 in binario.

Matrice generatrice (in forma sistemica):

$$G = [1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0]$$

$$[0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1]$$

$$[0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1]$$

$$[0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1]$$

Esempio di codifica:

$$m = [1 \ 0 \ 1 \ 1]$$

$$c = m \cdot G = [1 \ 0 \ 1 \ 1] \cdot G = [1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1]$$

$$\text{Verifica: } H \cdot c^T = 0$$

2.2.2 Decodifica a Sindrome

Definizione 2.5: La sindrome di un vettore ricevuto r è:

$$s = H \cdot r^T \in \text{GF}(q)^{(n-k)}$$

Proprietà della sindrome:

$$s = 0 \Leftrightarrow r \in C$$

$$s = H \cdot e^T \text{ dove } e = r - c \text{ è il vettore di errore}$$

Due vettori hanno la stessa sindrome $\Leftrightarrow$ sono nello stesso coset

Algoritmo di decodifica per codici di Hamming:

Input: r ricevuto

Output: $\tilde{c}$ decodificato

Calcola $s = H \cdot r^T$

Se $s = 0$: return r (nessun errore)

Altrimenti: trova la colonna h_j di H tale che $h_j = s$
 Correggi l'errore nella posizione j : $r_j \rightarrow r_j + s_{\text{peso}}$
 Return r corretto

Esempio 2.2: Decodifica con $\text{Ham}(3,2)$

Ricevuto: $r = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1]$

$s = H \cdot r^T = [0] \ [0]$

$[1] + [1] = [0]$

$[1] \ [0] \ [1]$

Risultato: $s = [0 \ 0 \ 1]^T = \text{colonna 1 di } H$

Errore in posizione 1. Correggiamo:

$\tilde{c} = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1]$ (flip del primo bit)

Verifica: $H \cdot \tilde{c}^T = 0 \checkmark$

2.2.3 Codici di Hamming Estesi

Definizione 2.6: Il codice di Hamming esteso $\hat{\text{ham}}(r,q)$ si ottiene aggiungendo un bit di parità globale:

$[\hat{n}, \hat{k}, \hat{d}] = [(q^r-1)/(q-1)+1, (q^r-1)/(q-1)-r, 4]$

Per $q=2$: $[2^r, 2^r-r-1, 4]$

Matrice di controllo:

$\hat{H} = [H \mid 1]$

$[1 \ 1 \ \dots \ 1 \mid 1]$

Dove la seconda riga impone che $\sum_i c_i = 0$.

Proprietà:

Il codice esteso è autoduale quando r è dispari

Raggiunge il limite di Plotkin per $d=4$

2.3 Codici di Golay

2.3.1 Codice di Golay Binario $[23,12,7]$

Il codice di Golay binario G_{23} è un codice perfetto con parametri $[23,12,7]$.

Costruzione tramite matrice generatrice:

$G = [I_{12} \mid A]$

dove A è una matrice 12×11 circolante con prima riga:

$a = [1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0]$

Le righe successive si ottengono per shift ciclico.

Matrice A esplicita:

$A = [1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0]$

$[0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1]$

$[1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0]$

$[0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0]$

$[0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0]$

[0 0 0 1 0 1 1 0 1 1 1]
 [1 0 0 0 1 0 1 1 0 1 1]
 [1 1 0 0 0 1 0 1 1 0 1]
 [1 1 1 0 0 0 1 0 1 1 0]
 [0 1 1 1 0 0 0 1 0 1 1]
 [1 0 1 1 1 0 0 0 1 0 1]
 [1 1 0 1 1 1 0 0 0 1 0]

Proprietà notevoli:

Codice perfetto: raggiunge il limite di Hamming

Può correggere $t = 3$ errori

Numero di parole codificate: $M = 2^{12} = 4096$

Verifica: $V_2(23,3) = 1 + 23 + 253 + 1771 = 2048$

$M \cdot V_2(23,3) = 4096 \cdot 2048 = 2^{23} \checkmark$

Codice di Golay esteso [24,12,8]:

Aggiungendo un bit di parità globale: G_{24} con parametri [24,12,8].

Proprietà:

Codice autoduale: $G_{24} = G_{24}^\perp$

Codice MDS: $8 = 24 - 12 + 1 - 3$ (quasi MDS)

Gruppo di automorfismi: gruppo di Mathieu M_{24}

2.3.2 Codice di Golay Ternario [11,6,5]

Codice perfetto su $GF(3)$ con parametri [11,6,5].

Matrice generatrice:

$G = [I_6 \mid B]$

dove B è una matrice 6×5 con elementi in $\{0,1,2\}$.

Verifica della perfezione:

$V_3(11,2) = \sum_{i=0}^2 \binom{11}{i} 2^i$

$= 1 + 11 \cdot 2 + 55 \cdot 4$

$= 1 + 22 + 220$

$= 243 = 3^5$

$M = 3^6 = 729$

$M \cdot V_3(11,2) = 729 \cdot 243 = 177147 = 3^{11} \checkmark$

2.4 Pesi e Distribuzioni di Peso

2.4.1 Enumeratore dei Pesi

Definizione 2.7: L'enumeratore dei pesi di un codice C è il polinomio:

$W_C(x,y) = \sum_{c \in C} x^{n-w_H(c)} \cdot y^{w_H(c)}$

Forma equivalente:

$$W_C(x,y) = \sum_{i=0}^n A_i \cdot x^{n-i} \cdot y^i$$

dove $A_i = |\{c \in C : w_H(c) = i\}|$ è il numero di parole di peso i .

Vettore di distribuzione dei pesi:

$$(A_0, A_1, A_2, \dots, A_n)$$

dove $A_0 = 1$ per codici lineari.

Esempio 2.3: Codice di ripetizione $[3,1,3]$ su $GF(2)$

$$C = \{000, 111\}$$

$$\text{Distribuzione: } (A_0, A_1, A_2, A_3) = (1, 0, 0, 1)$$

$$W_C(x,y) = x^3 + y^3$$

2.4.2 Identità di MacWilliams

Teorema 2.3 (MacWilliams Identity):

Sia C un codice lineare $[n,k]$ su $GF(q)$ e $C^\perp$ il suo codice duale. Allora:

$$W_{(C^\perp)}(x,y) = (1/|C|) \cdot W_C(x + (q-1)y, x - y)$$

Per $q=2$:

$$W_{(C^\perp)}(x,y) = (1/2^k) \cdot W_C(x + y, x - y)$$

Dimostrazione (schema):

Usa la teoria dei caratteri su gruppi abeliani finiti. Il carattere $\chi_a(c) = (-1)^{a \cdot c}$ per codici binari soddisfa:

$$\sum_{c \in C} \chi_a(c) = \begin{cases} |C| & \text{se } a \in C^\perp \\ 0 & \text{altrimenti} \end{cases}$$

Manipolando le somme su pesi e caratteri si ottiene l'identità. $\square$

Esempio 2.4: Codice di Hamming $[7,4,3]$

Distribuzione di peso di $\text{Ham}(3,2)$:

$$A_0 = 1$$

$$A_3 = 7$$

$$A_4 = 7$$

$$A_7 = 1$$

$$\text{Altri } A_i = 0$$

$$W_{\text{Ham}}(x,y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7$$

Codice duale $[7,3,4]$ (codice Simplex):

$$W_{(\text{Ham}^\perp)}(x,y) = (1/16)[x^7 + 7x^4y^3 + 7x^3y^4 + y^7] \text{ valutato in } (x+y, x-y)$$

$$= (1/16)[(x+y)^7 + 7(x+y)^4(x-y)^3 + 7(x+y)^3(x-y)^4 + (x-y)^7]$$

Calcolo esplicito:

$$W_{(\text{Ham}^\perp)}(x,y) = x^7 + 7y^4 \cdot x^3$$

$$\text{Distribuzione del Simplex: } (A_0, A_3, A_4, \dots) = (1, 0, 7, 0, \dots)$$

2.5 Codici Duali e Autoduali

2.5.1 Codice Duale

Definizione 2.8: Il codice duale $C^\perp$ di un codice lineare $C \subseteq GF(q)^n$ è:

$$C^\perp = \{x \in GF(q)^n : x \cdot c^T = 0 \text{ per ogni } c \in C\}$$

dove $x \cdot c^T = \sum_{i=1}^n x_i \cdot c_i$ è il prodotto scalare standard.

Teorema 2.4:

$$\dim(C) + \dim(C^\perp) = n$$

$$(C^\perp)^\perp = C$$

Se G è matrice generatrice di C , allora H^T è matrice generatrice di $C^\perp$

Se H è matrice di controllo di C , allora G^T è matrice di controllo di $C^\perp$

Dimostrazione di (1):

$C^\perp$ è l'ortogonale di C rispetto al prodotto scalare standard. Per il teorema dimensionale:

$$\dim(C) + \dim(C^\perp) = n \quad \square$$

2.5.2 Codici Autoduali

Definizione 2.9: Un codice C è autoduale se $C = C^\perp$.

Condizione necessaria: $n = 2k$ (la dimensione deve essere pari)

Classificazione:

Tipo I (binario): tutti i pesi sono pari

Tipo II (binario): tutti i pesi sono multipli di 4 (doppiamente pari)

Esempio 2.5: Codice di Golay esteso $[24,12,8]$

È autoduale di Tipo II. Distribuzione dei pesi:

$$A_0 = 1$$

$$A_8 = 759$$

$$A_{12} = 2576$$

$$A_{16} = 759$$

$$A_{24} = 1$$

$$W_{(G_{24})}(x,y) = x^{24} + 759x^{16}y^8 + 2576x^{12}y^{12} + 759x^8y^{16} + y^{24}$$

Teorema 2.5 (Gleason-Pierce):

L'enumeratore dei pesi di un codice binario autoduale di tipo II è un polinomio nel:

$$W_{(G_{24})}(x,y) \text{ e } W_{E_8}(x,y) = x^8 + 14x^4y^4 + y^8$$

dove E_8 è il reticolo di Gosset.

3. CODICI CICLICI

3.1 Struttura Algebrica

3.1.1 Definizione e Rappresentazione Polinomiale

Definizione 3.1: Un codice lineare $C \subseteq GF(q)^n$ è ciclico se per ogni parola codificata $c = (c_0, c_1, \dots, c_{(n-1)}) \in C$, anche lo shift ciclico:

$$\sigma(c) = (c_{(n-1)}, c_0, c_1, \dots, c_{(n-2)}) \in C$$

Rappresentazione polinomiale:

Associamo alla parola $c = (c_0, c_1, \dots, c_{(n-1)})$ il polinomio:

$$c(x) = c_0 + c_1 \cdot x + c_2 \cdot x^2 + \dots + c_{(n-1)} \cdot x^{(n-1)} \in \text{GF}(q)[x]$$

Proposizione 3.1: Lo shift ciclico $\sigma(c)$ corrisponde a:

$$\sigma(c)(x) \equiv x \cdot c(x) \pmod{x^n - 1}$$

Dimostrazione:

$$x \cdot c(x) = c_0 \cdot x + c_1 \cdot x^2 + \dots + c_{(n-1)} \cdot x^n$$

Poiché $x^n \equiv 1 \pmod{x^n - 1}$:

$$x \cdot c(x) \equiv c_{(n-1)} + c_0 \cdot x + c_1 \cdot x^2 + \dots + c_{(n-2)} \cdot x^{(n-1)} \pmod{x^n - 1}$$

che corrisponde a $\sigma(c)$. $\square$

Teorema 3.1 (Struttura algebrica dei codici ciclici):

I codici ciclici di lunghezza n su $\text{GF}(q)$ sono in corrispondenza biunivoca con gli ideali dell'anello quoziente:

$$R_n = \text{GF}(q)[x]/(x^n - 1)$$

Dimostrazione:

Un codice C è ciclico $\Leftrightarrow$ per ogni $c(x) \in C$ e ogni $p(x) \in \text{GF}(q)[x]$, si ha $p(x) \cdot c(x) \pmod{x^n - 1} \in C \Leftrightarrow C$ è un ideale di R_n . $\square$

3.1.2 Polinomio Generatore

Teorema 3.2 (Polinomio generatore):

Ogni codice ciclico non banale C è generato da un unico polinomio monico $g(x)$ di grado minimo, detto polinomio generatore, che soddisfa:

$$g(x) \mid (x^n - 1)$$

$$C = \{a(x) \cdot g(x) \pmod{x^n - 1} : a(x) \in \text{GF}(q)[x], \deg(a) \leq n - \deg(g) - 1\}$$

$$\dim(C) = n - \deg(g)$$

Dimostrazione:

Sia $g(x)$ il polinomio monico non nullo di grado minimo in C . Per ogni $c(x) \in C$, dividendo:

$$c(x) = q(x) \cdot g(x) + r(x), \deg(r) < \deg(g)$$

Si ha $r(x) = c(x) - q(x) \cdot g(x) \in C$ (chiusura). Per minimalità di $\deg(g)$, deve essere $r(x) = 0$, quindi $g(x) \mid c(x)$.

Ora, $g(x) \in C$ implica $x^i \cdot g(x) \pmod{x^n - 1} \in C$ per ogni i . In particolare, con i scelto tale che $\deg(x^i \cdot g(x)) \geq n$:

$$x^i \cdot g(x) = x^{(i-n)} \cdot (x^n - 1) + x^{(i-n)} + x^i \cdot g(x)$$

Quindi $x^n \cdot g(x) - g(x) = (x^n - 1) \cdot g(x) \in C$. Essendo C un ideale in R_n , segue $(x^n - 1) \in \langle g(x) \rangle$ come ideale, quindi $g(x) \mid (x^n - 1)$. $\square$

Notazione: Per un codice ciclico $[n, k]$ con generatore $g(x)$ di grado $n-k$:

$$g(x) = g_0 + g_1 \cdot x + \dots + g_{(n-k-1)} \cdot x^{(n-k-1)} + x^{(n-k)}$$

Matrice generatrice:

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \dots & g_{(n-k)} & 0 & \dots & 0 \\ 0 & g_0 & g_1 & \dots & g_{(n-k-1)} & g_{(n-k)} & \dots & 0 \\ 0 & 0 & g_0 & \dots & g_{(n-k-2)} & g_{(n-k-1)} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & g_0 & g_1 & \dots & g_{(n-k)} \end{bmatrix}$$

(matrice circolante)

3.1.3 Polinomio di Controllo

Definizione 3.2: Il polinomio di controllo (o polinomio di verifica) è:

$$h(x) = (x^n - 1)/g(x)$$

con $\deg(h) = k$.

Teorema 3.3:

Una parola $c(x)$ appartiene al codice ciclico generato da $g(x)$ se e solo se:

$$c(x) \cdot h(x) \equiv 0 \pmod{x^n - 1}$$

Dimostrazione:

$$c(x) = a(x) \cdot g(x) \text{ per qualche } a(x) \Rightarrow$$

$$c(x) \cdot h(x) = a(x) \cdot g(x) \cdot h(x) = a(x) \cdot (x^n - 1) \equiv 0 \pmod{x^n - 1} \quad \square$$

Matrice di controllo di parità:

$$H = \begin{bmatrix} h_k & h_{(k-1)} & \dots & h_1 & h_0 & 0 & \dots & 0 \\ 0 & h_k & \dots & h_2 & h_1 & h_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_k & h_{(k-1)} & \dots & h_1 & h_0 \end{bmatrix}$$

dove $h(x) = h_0 + h_1 \cdot x + \dots + h_k \cdot x^k$.

3.2 Fattorizzazione di $x^n - 1$

3.2.1 Polinomi Ciclotomici

Definizione 3.3: Per ogni intero positivo m , il m -esimo polinomio ciclotomico $Q_m(x)$ è:

$$Q_m(x) = \prod_{\zeta} (x - \zeta)$$

dove il prodotto è su tutte le radici m -esime primitive dell'unità ζ .

Grado: $\deg(Q_m) = \phi(m)$, dove ϕ è la funzione di Eulero.

Proprietà fondamentale:

$$x^n - 1 = \prod_{(m|n)} Q_m(x)$$

Esempi:

$$Q_1(x) = x - 1$$

$$Q_2(x) = x + 1$$

$$Q_3(x) = x^2 + x + 1$$

$$Q_4(x) = x^2 + 1$$

$$Q_5(x) = x^4 + x^3 + x^2 + x + 1$$

$$Q_6(x) = x^2 - x + 1$$

$$Q_7(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

$$Q_8(x) = x^4 + 1$$

Fattorizzazione su GF(2):

$$x^7 - 1 = (x - 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

$$x^{15} - 1 = (x - 1)(x^2 + x + 1)(x^4 + x + 1)(x^4 + x^3 + 1)(x^4 + x^3 + x^2 + x + 1)$$

Teorema 3.4: Su GF(q), il polinomio $x^n - 1$ si fattorizza come:

$$x^n - 1 = \prod_{i=1}^s f_i(x)$$

dove $f_i(x)$ sono polinomi irriducibili monici distinti.

3.2.2 Classi Ciclotomiche

Definizione 3.4: Per q e n coprimi, la q-classe ciclotomica di s modulo n è:

$$C_s = \{s, s \cdot q, s \cdot q^2, s \cdot q^3, \dots\} \pmod{n}$$

contenente tutti i valori distinti di $s \cdot q^i \pmod{n}$.

Teorema 3.5: I fattori irriducibili di $x^n - 1$ su GF(q) corrispondono alle classi ciclotomiche modulo n.

Ogni fattore irriducibile $f_s(x)$ ha grado $|C_s|$ e ha come radici $\{\alpha^j : j \in C_s\}$, dove α è una radice n-esima primitiva dell'unità in un'estensione di GF(q).

Esempio 3.1: Fattorizzazione di $x^7 - 1$ su GF(2)

$$n = 7, q = 2$$

Classi ciclotomiche modulo 7:

$$C_0 = \{0\}$$

$$C_1 = \{1, 2, 4\} \text{ (poiché } 1 \cdot 2 = 2, 2 \cdot 2 = 4, 4 \cdot 2 = 8 \equiv 1 \pmod{7})$$

$$C_3 = \{3, 6, 5\} \text{ (poiché } 3 \cdot 2 = 6, 6 \cdot 2 = 12 \equiv 5, 5 \cdot 2 = 10 \equiv 3 \pmod{7})$$

Quindi:

$$f_0(x) = x - 1 \text{ (grado 1)}$$

$$f_1(x) = \text{polinomio di grado 3 con radici } \alpha, \alpha^2, \alpha^4$$

$$f_3(x) = \text{polinomio di grado 3 con radici } \alpha^3, \alpha^5, \alpha^6$$

Esplicitamente:

$$f_1(x) = x^3 + x + 1$$

$$f_3(x) = x^3 + x^2 + 1$$

$$\text{Verifica: } (x-1)(x^3+x+1)(x^3+x^2+1) = x^7 - 1 \quad \checkmark$$

3.3 Codici BCH

3.3.1 Definizione e Costruzione

Definizione 3.5: Un codice BCH (Bose-Chaudhuri-Hocquenghem) di lunghezza n , distanza progettata δ e radice $\alpha \in GF(q^m)$ è un codice ciclico il cui polinomio generatore ha come radici:

$$\alpha^b, \alpha^{(b+1)}, \alpha^{(b+2)}, \dots, \alpha^{(b+\delta-2)}$$

per qualche intero $b \geq 0$ (tipicamente $b = 1$).

Polinomio generatore:

$$g(x) = \text{lcm}\{m_{\alpha^b}(x), m_{\alpha^{(b+1)}}(x), \dots, m_{\alpha^{(b+\delta-2)}}(x)\}$$

dove $m_{\beta}(x)$ è il polinomio minimo di β su $GF(q)$.

Teorema 3.6 (BCH Bound):

Un codice BCH con distanza progettata δ ha distanza minima $d \geq \delta$.

Dimostrazione:

Sia $c(x)$ una parola codificata non nulla di peso $w \leq \delta-1$. Possiamo scrivere:

$$c(x) = \sum_{j \in S} c_j \cdot x^j$$

dove $S \subseteq \{0, 1, \dots, n-1\}$ con $|S| = w \leq \delta-1$.

Per costruzione, $c(\alpha^i) = 0$ per $i = b, b+1, \dots, b+\delta-2$. Quindi:

$$\sum_{j \in S} c_j \cdot (\alpha^i)^j = 0, \text{ per } i = b, b+1, \dots, b+w-1$$

Questo è un sistema lineare omogeneo:

$$\begin{bmatrix} \alpha^{(j_1 \cdot b)} & \alpha^{(j_2 \cdot b)} & \dots & \alpha^{(j_w \cdot b)} \end{bmatrix} \begin{bmatrix} c_{(j_1)} \\ c_{(j_2)} \\ \vdots \\ c_{(j_w)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \alpha^{(j_1 \cdot (b+1))} & \alpha^{(j_2 \cdot (b+1))} & \dots & \alpha^{(j_w \cdot (b+1))} \end{bmatrix} \begin{bmatrix} c_{(j_1)} \\ c_{(j_2)} \\ \vdots \\ c_{(j_w)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} \alpha^{(j_1 \cdot (b+w-1))} & \alpha^{(j_2 \cdot (b+w-1))} & \dots & \alpha^{(j_w \cdot (b+w-1))} \end{bmatrix} \begin{bmatrix} c_{(j_1)} \\ c_{(j_2)} \\ \vdots \\ c_{(j_w)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

La matrice è di tipo Vandermonde:

$$V = [\alpha^{(j_i \cdot (b+k))}]_{(k=0, \dots, w-1; i=1, \dots, w)}$$

Il suo determinante è:

$$\det(V) = \alpha^{(b \cdot \sum_i j_i)} \cdot \prod_{(1 \leq i < l \leq w)} (\alpha^{(j_l)} - \alpha^{(j_i)}) \neq 0$$

Quindi l'unico vettore soluzione è $c = 0$, contraddizione. Dunque $w \geq \delta$. $\square$

3.3.2 Codici BCH Primitivi

Definizione 3.6: Un codice BCH primitivo ha lunghezza $n = q^m - 1$, dove α è una radice n -esima primitiva dell'unità in $GF(q^m)$.

Parametri: $[q^m - 1, \geq q^m - 1 - m \cdot (\delta - 1), \geq \delta]$

Per $q = 2$ (codici binari):

$$[n, k, d] \text{ con } n = 2^m - 1, k \geq n - m \cdot (\delta - 1), d \geq \delta$$

Esempio 3.2: Codice BCH binario $[15, 7, 5]$

Parametri: $m = 4, n = 15, \delta = 5, b = 1$

Campo: $GF(2^4)$ generato da α con polinomio primitivo $x^4 + x + 1$.

Polinomio generatore con radici $\alpha, \alpha^2, \alpha^3, \alpha^4$:

$$g(x) = \text{lcm}\{m_{\alpha}(x), m_{\alpha^2}(x), m_{\alpha^3}(x), m_{\alpha^4}(x)\}$$

Classi ciclotomiche modulo 15:

$$C_1 = \{1, 2, 4, 8\}$$

$$C_3 = \{3, 6, 12, 9\}$$

$$C_5 = \{5, 10\}$$

$$C_7 = \{7, 14, 13, 11\}$$

$$m_{\alpha}(x) = m_{\alpha^2}(x) = m_{\alpha^4}(x) = m_{\alpha^8}(x) = x^4 + x + 1$$

$$m_{\alpha^3}(x) = m_{\alpha^6}(x) = m_{\alpha^9}(x) = m_{\alpha^{12}}(x) = x^4 + x^3 + x^2 + x + 1$$

$$g(x) = (x^4 + x + 1)(x^4 + x^3 + x^2 + x + 1)$$

$$= x^8 + x^7 + x^6 + x^4 + 1$$

$$\text{Dimensione: } k = 15 - 8 = 7 \checkmark$$

3.3.3 Algoritmo di Decodifica di Peterson-Gorenstein-Zierler

Sindrome:

Per un vettore ricevuto $r(x)$, le componenti della sindrome sono:

$$S_i = r(\alpha^i) = \sum_{j=0}^{n-1} r_j \cdot \alpha^{i \cdot j}, \text{ per } i = 1, 2, \dots, 2t$$

dove $t = \lfloor (\delta-1)/2 \rfloor$ è il numero massimo di errori correggibili.

Vettore di errore:

$$e(x) = \sum_{j \in E} e_j \cdot x^j$$

dove E è l'insieme delle posizioni di errore.

Locatori e valori di errore:

Per ogni posizione di errore $i \in E$:

$$X_i = \alpha^i \text{ è il locatore di errore}$$

$$Y_i = e_i \text{ è il valore di errore}$$

Sistema di equazioni sindromiche:

$$S_i = \sum_{j \in E} Y_j \cdot X_j^i, \text{ per } i = 1, 2, \dots, 2t$$

Polinomio locatore di errori:

$$\Lambda(x) = \prod_{i \in E} (1 - X_i \cdot x) = 1 + \Lambda_1 \cdot x + \Lambda_2 \cdot x^2 + \dots + \Lambda_v \cdot x^v$$

dove $v = |E|$ è il numero di errori.

Equazioni chiave (Newton's identities):

$$S_i + \Lambda_1 \cdot S_{i-1} + \Lambda_2 \cdot S_{i-2} + \dots + \Lambda_v \cdot S_{i-v} = 0$$

per $i = v+1, \dots, 2t$.

Algoritmo di Peterson:

Calcola le sindromi $S_1, S_2, \dots, S_{2t}$

Costruisci la matrice di sindrome:

$$M = \begin{bmatrix} S_1 & S_2 & \dots & S_v \\ S_2 & S_3 & \dots & S_{v+1} \end{bmatrix}$$

$$[S_2 \quad S_3 \quad \dots \quad S_{v+1}]$$

$$\begin{bmatrix} 1 & \alpha^3 & \alpha^6 & \alpha^9 \\ 1 & \alpha^6 & \alpha^{12} & \alpha^{15} \end{bmatrix}$$

$$[S_1 \ S_2 \ S_3 \ S_4]$$

Trova v massimo tale che $\det(M) \neq 0$

Risolvi il sistema lineare:

$$[S_1 \ S_2 \ S_3 \ S_4] [\Lambda_1 \ \Lambda_2] = [S_5]$$

$$[S_2 \ S_3 \ S_4 \ S_5] [\Lambda_2 \ \Lambda_3] = [S_6]$$

$$\begin{bmatrix} 1 & \alpha^3 & \alpha^6 & \alpha^9 \\ 1 & \alpha^6 & \alpha^{12} & \alpha^{15} \end{bmatrix} \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \end{bmatrix} = \begin{bmatrix} S_5 \\ S_6 \end{bmatrix}$$

$$[S_1 \ S_2 \ S_3 \ S_4] [\Lambda_1 \ \Lambda_2] = [S_5]$$

con segno negativo.

$$\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2$$

Trova le radici di $\Lambda(x)$: se X_i è radice, allora α^{-i} indica posizione di errore

Calcola i valori di errore Y_i risolvendo:

$$[X_1 \ X_2 \ X_3 \ X_4] [Y_1 \ Y_2] = [S_5]$$

$$[X_1^2 \ X_2^2 \ X_3^2 \ X_4^2] [Y_2] = [S_6]$$

$$\begin{bmatrix} 1 & \alpha^3 & \alpha^6 & \alpha^9 \\ 1 & \alpha^6 & \alpha^{12} & \alpha^{15} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} S_5 \\ S_6 \end{bmatrix}$$

$$[X_1^3 \ X_2^3 \ X_3^3 \ X_4^3] [Y_1 \ Y_2] = [S_7]$$

$$\hat{c}(x) = r(x) - e(x)$$

Esempio 3.3: Decodifica BCH [15,7,5]

Supponiamo di ricevere $r(x)$ con 2 errori nelle posizioni 3 e 7.

$$e(x) = x^3 + x^7$$

Sindromi:

$$S_1 = r(\alpha) = \alpha^3 + \alpha^7$$

$$S_2 = r(\alpha^2) = \alpha^6 + \alpha^{14}$$

$$S_3 = r(\alpha^3) = \alpha^9 + \alpha^{21} = \alpha^9 + \alpha^6 \text{ (poiché } \alpha^{15} = 1)$$

$$S_4 = r(\alpha^4) = \alpha^{12} + \alpha^{28} = \alpha^{12} + \alpha^{13}$$

Matrice per $v = 2$:

$$M = [S_1 \ S_2] \begin{bmatrix} \alpha^3 + \alpha^7 & \alpha^6 + \alpha^{14} \end{bmatrix}$$

$$[S_2 \ S_3] = [\alpha^6 + \alpha^{14} \ \alpha^9 + \alpha^6]$$

Sistema:

$$[\alpha^3 + \alpha^7 \ \alpha^6 + \alpha^{14}] [\Lambda_1 \ \Lambda_2] = [\alpha^9 + \alpha^6]$$

$$[\Lambda_1] = [\alpha^{12} + \alpha^{13}]$$

Risolvendo (calcoli in $GF(2^4)$):

$$\Lambda_2 = \alpha^{10}, \Lambda_1 = \alpha^3 + \alpha^7$$

$$\Lambda(x) = 1 + (\alpha^3 + \alpha^7) \cdot x + \alpha^{10} \cdot x^2$$

$$\text{Radici: } \alpha^{-3} \text{ e } \alpha^{-7} \text{ (cioè } \alpha^{12} \text{ e } \alpha^8)$$

Posizioni di errore: 3 e 7 ✓

3.4 Codici Reed-Solomon

3.4.1 Definizione e Proprietà

Definizione 3.7: Un codice Reed-Solomon $RS(n,k)$ è un codice BCH non binario su $GF(q)$ con:

Lunghezza: $n \leq q - 1$

Dimensione: k

Distanza minima: $d = n - k + 1$ (MDS)

Costruzione: Prendendo un codice BCH primitivo su $GF(q)$ con $\delta = n - k + 1$, si ottiene un codice con distanza minima esattamente $d = n - k + 1$.

Polinomio generatore:

$$g(x) = \prod_{i=1}^{n-k} (x - \alpha^i) = g_0 + g_1 \cdot x + \dots + g_{n-k-1} \cdot x^{n-k-1} + x^{n-k}$$

dove α è primitivo in $GF(q)$.

Proprietà MDS: I codici Reed-Solomon raggiungono il limite di Singleton:

$$k = n - d + 1$$

Equivalentemente: $d = n - k + 1$

Teorema 3.7: Ogni sottoinsieme di k colonne della matrice di controllo di parità di un codice Reed-Solomon è linearmente indipendente.

Matrice di controllo di parità:

$$H = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^n \\ \alpha^2 & \alpha^4 & \alpha^6 & \dots & \alpha^{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{n-k-1} & \alpha^{2(n-k-1)} & \alpha^{3(n-k-1)} & \dots & \alpha^{n(n-k-1)} \end{bmatrix}$$

$$[\alpha \quad \alpha^2 \quad \alpha^3 \quad \dots \quad \alpha^n]$$

$$[\alpha^2 \quad \alpha^4 \quad \alpha^6 \quad \dots \quad \alpha^{2n}]$$

$$[\vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots]$$

$$[\alpha^{n-k-1} \quad \alpha^{2(n-k-1)} \quad \alpha^{3(n-k-1)} \quad \dots \quad \alpha^{n(n-k-1)}]$$

Questa è una matrice di Vandermonde, e qualsiasi k colonne hanno determinante non nullo.

3.4.2 Codifica Sistemica

Metodo 1: Divisione polinomiale

Per messaggio $m = (m_0, m_1, \dots, m_{k-1})$, formiamo:

$$m(x) = m_0 + m_1 \cdot x + \dots + m_{k-1} \cdot x^{k-1}$$

Calcoliamo:

$$x^{n-k} \cdot m(x) = q(x) \cdot g(x) + r(x)$$

dove $\deg(r) < n - k$.

La parola codificata è:

$$c(x) = x^{n-k} \cdot m(x) - r(x) = q(x) \cdot g(x)$$

In forma vettoriale:

$$c = [r_0, r_1, \dots, r_{n-k-1}, m_0, m_1, \dots, m_{k-1}]$$

Esempio 3.4: RS(7,3) su $GF(8)$

$GF(8)$ generato da α con $\alpha^3 + \alpha + 1 = 0$.

Elementi: $\{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$$

Espandendo in $GF(8)$:

$$g(x) = x^4 + \alpha^3 \cdot x^3 + x^2 + \alpha \cdot x + \alpha^3$$

Messaggio: $m = (1, \alpha, \alpha^2) \rightarrow m(x) = 1 + \alpha \cdot x + \alpha^2 \cdot x^2$

$$x^4 \cdot m(x) = x^4 + \alpha \cdot x^5 + \alpha^2 \cdot x^6$$

Divisione:

$$x^4 + \alpha \cdot x^5 + \alpha^2 \cdot x^6 = q(x) \cdot g(x) + r(x)$$

Calcolando:

$$q(x) = \alpha^2 \cdot x^2 + \alpha^4 \cdot x + \alpha$$

$$r(x) = \alpha^6 \cdot x^3 + \alpha^5 \cdot x^2 + \alpha^3 \cdot x + \alpha^2$$

Parola codificata:

$$c(x) = \alpha^2 + \alpha^3 \cdot x + \alpha^5 \cdot x^2 + \alpha^6 \cdot x^3 + x^4 + \alpha \cdot x^5 + \alpha^2 \cdot x^6$$

$$c = (\alpha^2, \alpha^3, \alpha^5, \alpha^6, 1, \alpha, \alpha^2)$$

3.4.3 Decodifica di Reed-Solomon

Algoritmo di Berlekamp-Massey:

Più efficiente dell'algoritmo di Peterson per trovare $\Lambda(x)$.

Input: Sindromi $S_1, S_2, \dots, S_{2t}$

Output: Polinomio locatore di errori $\Lambda(x)$

Inizializzazione:

$$\Lambda^{(0)}(x) = 1$$

$$B^{(0)}(x) = 1$$

$$L_0 = 0 \text{ (grado corrente)}$$

$$b_0 = 1$$

Iterazione $r = 1, \dots, 2t$:

Calcola discrepanza:

$$\Delta_r = S_r + \sum_{i=1}^{L_{r-1}} \Lambda_i^{(r-1)} \cdot S_{r-i}$$

Se $\Delta_r = 0$:

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x)$$

$$B^{(r)}(x) = x \cdot B^{(r-1)}(x)$$

$$L_r = L_{r-1}$$

$$b_r = b_{r-1}$$

Se $\Delta_r \neq 0$:

$$T(x) = \Lambda^{(r-1)}(x) - (\Delta_r / b_{r-1}) \cdot x \cdot B^{(r-1)}(x)$$

Se $2L_{r-1} \leq r - 1$:

$$\Lambda^{(r)}(x) = T(x)$$

$$B^{(r)}(x) = \Lambda^{(r-1)}(x)$$

$$L_r = r - L_{r-1}$$

$$b_r = \Delta_r$$

Altrimenti:

$$\Lambda^r(x) = T(x)$$

$$B^r(x) = x \cdot B^{r-1}(x)$$

$$L_r = L_{r-1}$$

$$b_r = b_{r-1}$$

$$\text{Output: } \Lambda(x) = \Lambda^{(2t)}(x)$$

Complessità: $O(t^2)$ operazioni in $GF(q)$, contro $O(t^3)$ di Peterson.

Algoritmo di Chien: Ricerca esaustiva delle radici

Per trovare le radici di $\Lambda(x) = \sum_{i=0}^v \Lambda_i \cdot x^i$:

For $j = 0$ to $n-1$:

Valuta $\Lambda(\alpha^{(-j)})$

If $\Lambda(\alpha^{(-j)}) = 0$:

Posizione di errore in j

Algoritmo di Forney: Calcolo dei valori di errore

Definisci il polinomio valutatore di errore:

$$\Omega(x) = S(x) \cdot \Lambda(x) \bmod x^{(2t+1)}$$

dove $S(x) = S_1 \cdot x + S_2 \cdot x^2 + \dots + S_{(2t)} \cdot x^{(2t)}$.

Il valore di errore nella posizione j (con locatore $X_j = \alpha^j$) è:

$$Y_j = -\Omega(X_j^{(-1)}) / \Lambda'(X_j^{(-1)})$$

dove $\Lambda'(x)$ è la derivata formale di $\Lambda(x)$.

Esempio 3.5: Decodifica RS(7,3) con 2 errori

Ricevuto: r con errori in posizioni 2 e 5, valori α^3 e α^2 rispettivamente.

Sindromi:

$$S_1 = \alpha^3 \cdot \alpha^2 + \alpha^2 \cdot \alpha^5 = \alpha^5 + \alpha^7 = \alpha^5 + 1$$

$$S_2 = \alpha^3 \cdot \alpha^4 + \alpha^2 \cdot \alpha^{10} = \alpha^7 + \alpha^{12} = 1 + \alpha^5$$

$$S_3 = \alpha^3 \cdot \alpha^6 + \alpha^2 \cdot \alpha^{15} = \alpha^9 + \alpha^{17} = \alpha^2 + \alpha^3$$

$$S_4 = \alpha^3 \cdot \alpha^8 + \alpha^2 \cdot \alpha^{20} = \alpha^{11} + \alpha^{22} = \alpha^4 + 1$$

Berlekamp-Massey produce:

$$\Lambda(x) = 1 + \alpha^4 \cdot x + \alpha^3 \cdot x^2$$

Chien search:

$$\Lambda(\alpha^{(-2)}) = 1 + \alpha^4 \cdot \alpha^{(-2)} + \alpha^3 \cdot \alpha^{(-4)} = 1 + \alpha^2 + \alpha^{(-1)} = 0 \quad \checkmark$$

$$\Lambda(\alpha^{(-5)}) = 1 + \alpha^4 \cdot \alpha^{(-5)} + \alpha^3 \cdot \alpha^{(-10)} = 1 + \alpha^{(-1)} + \alpha^{(-7)} = 0 \quad \checkmark$$

Forney:

$$\begin{aligned} \Omega(x) &= (\alpha^5 + 1) \cdot x + (1 + \alpha^5) \cdot x^2 + (\alpha^2 + \alpha^3) \cdot x^3 + (\alpha^4 + 1) \cdot x^4 \cdot \Lambda(x) \bmod x^5 \\ &= \dots \text{ (calcolo)} \end{aligned}$$

$$Y_2 = -\Omega(\alpha^{-2}) / \Lambda'(\alpha^{-2})$$

$$\Lambda'(x) = \alpha^4 + 2\alpha^3 \cdot x = \alpha^4 \text{ (in caratteristica 2, il termine } 2\alpha^3 \cdot x = 0)$$

$$Y_2 = -\Omega(\alpha^{-2}) / \alpha^4 = \alpha^3 \checkmark$$

$$Y_5 = -\Omega(\alpha^{-5}) / \Lambda'(\alpha^{-5}) = \alpha^2 \checkmark$$

Correzione: $\hat{c} = r - e$ e dove e ha valore α^3 in posizione 2 e α^2 in posizione 5.

3.4.4 Applicazioni dei Codici Reed-Solomon

Sistemi di comunicazione:

CD Audio: RS(32,28) su GF(2⁸)

DVD: RS(208,192) su GF(2⁸)

Comunicazioni spaziali: concatenazione RS + convoluzionale

DVB (Digital Video Broadcasting): RS(204,188)

Codici concatenati:

Schema di concatenazione esterna/interna:

Codice esterno: Reed-Solomon

Codice interno: Convoluzionale o LDPC

Guadagno: Il codice esterno corregge burst errors lasciati dal decodificatore interno.

Parametri tipici:

RS esterno: (255, 223) su GF(2⁸), corregge 16 byte errors

Codice interno: rate 1/2 convoluzionale

4. CODICI CONVOLUZIONALI

4.1 Definizione e Struttura

4.1.1 Codificatore Convoluzionale

Definizione 4.1: Un codificatore convoluzionale (n,k,K) processa k bit di input alla volta e produce n bit di output, con memoria di K-1 stadi precedenti.

Parametri:

k: numero di bit di input per unità temporale

n: numero di bit di output per unità temporale

K: lunghezza del vincolo (constraint length)

Rate: $R = k/n$

Rappresentazione tramite polinomi generatori:

Per un codice (n,1,K), definiamo n polinomi generatori:

$$g_i(D) = g_{i,0} + g_{i,1} \cdot D + g_{i,2} \cdot D^2 + \dots + g_{i,K-1} \cdot D^{K-1}$$

per $i = 1, 2, \dots, n$, dove D è l'operatore di ritardo.

Sequenza di output:

$$v_i(D) = u(D) \cdot g_i(D)$$

dove $u(D) = u_0 + u_1 \cdot D + u_2 \cdot D^2 + \dots$ è la sequenza di input.

Esempio 4.1: Codificatore (2,1,3)

Polinomi generatori in ottale:

$$g_1(D) = 1 + D^2 = (1,0,1) = 5_8$$

$$g_2(D) = 1 + D + D^2 = (1,1,1) = 7_8$$

Notazione compatta: $[5, 7]_8$

Input: $u = 1\ 0\ 1\ 1\ 0\ 0\ \dots$

Output per g_1 : $v_1 = 1\ 0\ 1\ 0\ 1\ 1\ \dots$

Output per g_2 : $v_2 = 1\ 1\ 0\ 0\ 0\ 1\ \dots$

Multiplexed: $v = 11\ 01\ 10\ 00\ 01\ 11\ \dots$

4.1.2 Diagramma degli Stati

Rappresentazione a stati:

Stato $S_t \in \{0,1\}^{(K-1)}$ contiene i $K-1$ bit precedenti.

Transizioni:

$$S_{(t+1)} = f(S_t, u_t)$$

$$v_t = h(S_t, u_t)$$

dove:

f : funzione di transizione di stato

h : funzione di output

Esempio 4.2: Diagramma per codice $[5,7]_8$

Stati: $S = \{00, 01, 10, 11\}$

Transizioni (formato: input/output):

Da 00: --0/00--> 00

--1/11--> 10

Da 01: --0/11--> 00

--1/00--> 10

Da 10: --0/10--> 01

--1/01--> 11

Da 11: --0/01--> 01

--1/10--> 11

Matrice di transizione:

Ogni transizione da stato S_i a S_j con input u e output v contribuisce $D^{(w_H(v))} \cdot x^u$ al termine (i,j) .

Per $[5,7]_8$:

$$T(D,x) = \begin{bmatrix} D^0 x^0 + D^2 x^1 & 0 & & \\ & & & \end{bmatrix}$$

$$\begin{bmatrix} D^2x^0 + D^0x^1 & 0 \\ 0 & D^1x^0 + D^1x^1 \\ 0 & D^1x^0 + D^1x^1 \end{bmatrix}$$

4.1.3 Traliccio (Trellis)

Struttura temporale: Sviluppo del diagramma degli stati nel tempo.

Tempo: $t=0$ $t=1$ $t=2$ $t=3$...

```

00 ----00-->00---00-->00---00-->00
  \  11   \  11   \  11
   \       \       \
  10        10        10
10 ----01-->01---01-->01---01-->01
   10     10     10

```

...

Ogni percorso nel traliccio corrisponde a una sequenza codificata.

4.2 Distanza Libera

4.2.1 Definizione

Definizione 4.2: La distanza libera d_{free} di un codice convoluzionale è la minima distanza di Hamming tra due percorsi distinti nel traliccio che divergono e poi riconvergono.

$d_{\text{free}} = \min\{w_H(v) : v \neq 0 \text{ è una sequenza di output per qualche input}\}$

Teorema 4.1: Un codice convoluzionale con distanza libera d_{free} può correggere fino a:

$$t = \lfloor (d_{\text{free}} - 1)/2 \rfloor$$

errori in una finestra temporale sufficientemente lunga.

4.2.2 Funzione di Trasferimento

Definizione 4.3: La funzione di trasferimento $T(D,N,L)$ è la generatrice dei percorsi che divergono dallo stato zero e vi riconvergono, dove:

- D^w : peso di Hamming w dell'output
- N^k : input di peso k
- L^l : lunghezza l del percorso

Calcolo: Eliminando lo stato zero dal grafo degli stati e risolvendo le equazioni dei nodi.

Esempio 4.3: Funzione di trasferimento per $[5,7]_8$

Equazioni dei nodi (con stato 00 come riferimento):

$$X_{10} = D^2 \cdot N \cdot L + D \cdot L \cdot X_{11}$$

$$X_{01} = D \cdot L \cdot X_{10} + D \cdot L \cdot X_{11}$$

$$X_{11} = D \cdot L \cdot X_{10} + D \cdot L \cdot X_{01}$$

Risolvendo:

$$X_{10} = D^2 \cdot N \cdot L / (1 - D \cdot L \cdot (X_{11}/X_{10}))$$

Dopo calcoli algebrici:

$$T(D, N, L) = D^5 \cdot N \cdot L^3 / (1 - 2D \cdot N \cdot L)$$

Ponendo $N = L = 1$ per contare solo il peso:

$$T(D) = D^5 / (1 - 2D) = D^5 \cdot \sum_{i=0}^{\infty} (2D)^i = \sum_{i=0}^{\infty} 2^i \cdot D^{5+i}$$

Interpretazione:

- $d_{\text{free}} = 5$ (coefficiente del termine di grado minimo)
- Esistono 1 percorso di peso 5
- Esistono 2 percorsi di peso 6
- Esistono 4 percorsi di peso 7
- etc.

4.3 Algoritmo di Viterbi

4.3.1 Principio

Programmazione dinamica: Trova il percorso di massima verosimiglianza nel traliccio.

Metrica: Per un canale BSC con probabilità di errore p :

$$\text{Metrica di percorso} = \sum_t d_H(r_t, v_t)$$

dove r_t è ricevuto e v_t è l'output del percorso.

Principio di ottimalità: Il miglior percorso che termina nello stato S al tempo t passa attraverso il miglior percorso che termina in qualche stato S' al tempo $t-1$.

4.3.2 Algoritmo

Input: Sequenza ricevuta $r = r_0, r_1, r_2, \dots, r_L$

Output: Sequenza decodificata $\hat{u}$

Inizializzazione:

- $\mu_0(S) = 0$ se $S = 00$, altrimenti $\mu_0(S) = \infty$
- Predecessore $\pi_0(S) = \text{null}$

For $t = 1$ to L :

For ogni stato S :

$$\mu_t(S) = \min_{S'} \{ \mu_{t-1}(S') + d_H(r_{t-1}, \text{output}(S' \rightarrow S)) \}$$

$$\pi_t(S) = \operatorname{argmin}_{S'} \{ \mu_{t-1}(S') + d_H(r_{t-1}, \text{output}(S' \rightarrow S)) \}$$

Terminazione:

- $S = \operatorname{argmin}_S \mu_L(S)$
- Traceback: ricostruisci il percorso da S usando π_t

Complessità: $O(L \cdot |\text{Stati}| \cdot |\text{Transizioni}|) = O(L \cdot 2^{(K-1)} \cdot 2^k)$

Per rate $1/2$: $O(L \cdot 2^{(K-1)} \cdot 2) = O(L \cdot 2^K)$

4.3.3 Esempio Numerico

Esempio 4.4: Decodifica Viterbi per $[5,7]_8$

Ricevuto: $r = 11\ 10\ 01\ 11\ 01\ 10\ 00$

Stati: $\{00, 01, 10, 11\}$

$t=0$: $\mu(00) = 0$, altri $= \infty$

$t=1$: $r_0 = 11$

Transizioni possibili:

- $00 \rightarrow 00$: $\mu(00) = 0 + d_H(11, 00) = 0 + 2 = 2$
- $00 \rightarrow 10$: $\mu(10) = 0 + d_H(11, 11) = 0 + 0 = 0 \checkmark$

Metriche: $\mu_1(00)=2$, $\mu_1(10)=0$, altri $= \infty$

$t=2$: $r_1 = 10$

Da 00:

- 00 --0/00--> 00: $2 + d_H(10,00) = 2 + 1 = 3$
- 00 --1/11--> 10: $2 + d_H(10,11) = 2 + 1 = 3$

Da 10:

- 10 --0/10--> 01: $0 + d_H(10,10) = 0 + 0 = 0 \checkmark$
- 10 --1/01--> 11: $0 + d_H(10,01) = 0 + 1 = 1$

Metriche: $\mu_2(00)=3$, $\mu_2(01)=0$, $\mu_2(10)=3$, $\mu_2(11)=1$

t=3: $r_2 = 01$

Da 00:

- 00 --0/00--> 00: $3 + d_H(01,00) = 3 + 1 = 4$
- 00 --1/11--> 10: $3 + d_H(01,11) = 3 + 1 = 4$

Da 01:

- 01 --0/11--> 00: $0 + d_H(01,11) = 0 + 1 = 1 \checkmark$
- 01 --1/00--> 10: $0 + d_H(01,00) = 0 + 1 = 1$

Da 10:

- 10 --0/10--> 01: $3 + d_H(01,10) = 3 + 1 = 4$
- 10 --1/01--> 11: $3 + d_H(01,01) = 3 + 0 = 3$

Da 11:

- 11 --0/01--> 01: $1 + d_H(01,01) = 1 + 0 = 1$
- 11 --1/10--> 11: $1 + d_H(01,10) = 1 + 1 = 2$

Metriche: $\mu_3(00)=1$, $\mu_3(01)=1$, $\mu_3(10)=1$, $\mu_3(11)=2$

Continua fino a t=7, poi traceback...

Percorso ottimo: 00→10→01→00→10→01→00→00

Input decodificato: $u = 1\ 0\ 0\ 1\ 0\ 0\ 0$

4.4 Codici Punctured e Rate Compatible

4.4.1 Puncturing

Definizione 4.4: Il puncturing consiste nell'eliminare sistematicamente alcuni bit di output per aumentare il rate del codice.

Matrice di puncturing P: matrice binaria $n \times p$ dove:

- Righe: uscite del codificatore
- Colonne: periodi temporali

- 1: bit trasmesso, 0: bit eliminato

Rate modificato:

$$R' = k \cdot p / (n \cdot \sum_{(i,j)} P_{(i,j)})$$

Esempio 4.5: Puncturing di $[5,7]_8$

Codice madre: rate $1/2$

Matrice di puncturing:

$$P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Su 3 periodi:

- Periodo 1: trasmetti entrambi i bit $\rightarrow 11$
- Periodo 2: trasmetti solo bit 1 $\rightarrow 1_$
- Periodo 3: trasmetti solo bit 2 $\rightarrow _1$

Rate effettivo: $R' = 1 \cdot 3 / (2 \cdot 4) = 3/8$ (o equivalentemente $3/4$ considerando 3 bit input per 4 output)

Effetto sulla distanza libera:

$$d_{\text{free}}(\text{punctured}) \leq d_{\text{free}}(\text{originale})$$

La perdita dipende dalla matrice P scelta.

4.4.2 Rate Compatible Punctured Convolutional (RCPC)

Famiglia di codici: Da un codice madre a basso rate, si genera una famiglia di codici a rate crescente tramite puncturing progressivo.

Proprietà di compatibilità: I bit trasmessi da un codice a rate basso sono inclusi nei bit trasmessi da codici a rate più alto.

Applicazione: Adattamento del rate in sistemi ARQ (Automatic Repeat Request).

Esempio 4.6: Famiglia RCPC da rate $1/4$

Codice madre: $(4,1,K)$ con $g_1=1+D^2$, $g_2=1+D+D^2$, $g_3=1$, $g_4=1+D$

Matrici di puncturing:

Rate 1/4: $P_{1/4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Rate 1/3: $P_{1/3} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

Rate 1/2: $P_{1/2} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$

Rate 3/4: $P_{3/4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

5. CODICI LDPC E TURBO

5.1 Codici LDPC (Low-Density Parity-Check)

5.1.1 Definizione e Struttura

Definizione 5.1: Un codice LDPC è un codice lineare definito da una matrice di controllo di parità sparsa (con pochi elementi non-nulli).

Parametri di sparsità:

- d_v : grado delle colonne (numero di 1 per colonna)
- d_c : grado delle righe (numero di 1 per riga)
- Codice (d_v, d_c) -regolare se tutti i gradi sono costanti

Matrice di controllo H : $(n-k) \times n$ con:

- Densità: $\rho = (\text{numero di } 1) / ((n-k) \cdot n) \ll 1$
- Tipicamente: $d_v \in [2, 10]$, $d_c \in [6, 30]$

Grafo di Tanner: Rappresentazione bipartita di H

- Nodi variabili: n nodi (uno per colonna di H)

- Nodi di controllo: $n-k$ nodi (uno per riga di H)
- Arco tra v_i e c_j se $H_{(j,i)} = 1$

Esempio 5.1: LDPC (3,6)-regolare con $n=8$, $k=4$

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Ogni colonna ha peso 2, ogni riga ha peso 4 (approssimando (3,6)).

5.1.2 Costruzione di Codici LDPC

Metodo 1: Costruzione aleatoria

1. Fissa n , k , d_v , d_c
2. Genera matrice H random con vincoli di grado
3. Verifica assenza di cicli corti ($\text{girth} \geq 6$)
4. Verifica che $\text{rank}(H) = n-k$

Metodo 2: Costruzione algebrica - Array LDPC

Basata su Latin Squares e campi finiti.

Per p primo e $q = p^s$:

$$H = \begin{bmatrix} H_{0,0} & H_{0,1} & \dots & H_{0,p-1} \\ H_{1,0} & H_{1,1} & \dots & H_{1,p-1} \\ \vdots & \vdots & \ddots & \vdots \\ H_{(p-1),0} & H_{(p-1),1} & \dots & H_{(p-1),p-1} \end{bmatrix}$$

dove $H_{(i,j)}$ è una matrice $p \times p$ permutazione circolante.

Metodo 3: Codici QC-LDPC (Quasi-Cyclic)

Matrice a blocchi circolanti:

$$H = \begin{bmatrix} P^{(a_{0,0})} & P^{(a_{0,1})} & \dots & P^{(a_{0,m-1})} \\ P^{(a_{1,0})} & P^{(a_{1,1})} & \dots & P^{(a_{1,m-1})} \\ \vdots & \vdots & \ddots & \vdots \\ P^{(a_{l-1,0})} & P^{(a_{l-1,1})} & \dots & P^{(a_{l-1,m-1})} \end{bmatrix}$$

dove P è la matrice di permutazione ciclica $p \times p$ e $P^\infty =$ matrice nulla.

Esempio 5.2: QC-LDPC con $p=5$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$P^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$H = \begin{bmatrix} P^1 & P^\infty & P^2 \\ P^\infty & P^2 & P^4 \end{bmatrix}$$

fornisce un codice $[15,5]$ LDPC.

5.1.3 Algoritmo Sum-Product (Belief Propagation)

Decodifica iterativa basata su messaggi nel grafo di Tanner.

Messaggi:

- $\alpha_{(v \rightarrow c)}$: da nodo variabile v a nodo controllo c (log-likelihood ratio)
- $\beta_{(c \rightarrow v)}$: da nodo controllo c a nodo variabile v

Inizializzazione: Per canale AWGN con input r_i :

$$\alpha_i^{(0)} = (2 \cdot r_i) / \sigma^2$$

dove σ^2 è la varianza del rumore.

Iterazione l :

1. Update nodi controllo (check node update):

$$\beta_{(c \rightarrow v)}^{(l)} = 2 \cdot \operatorname{arctanh} \left(\prod_{(v' \in N(c) \setminus v)} \tanh(\alpha_{(v' \rightarrow c)}^{(l-1)} / 2) \right)$$

dove $N(c)$ sono i vicini di c nel grafo.

Approssimazione min-sum (più semplice):

$$\beta_{(c \rightarrow v)}^{(l)} \approx \left(\prod_{(v' \in N(c) \setminus v)} \text{sign}(\alpha_{(v' \rightarrow c)}^{(l-1)}) \right) \cdot \min_{(v' \in N(c) \setminus v)} |\alpha_{(v' \rightarrow c)}^{(l-1)}|$$

2. Update nodi variabili (variable node update):

$$\alpha_{(v \rightarrow c)}^{(l)} = (2 \cdot r_v) / \sigma^2 + \sum_{(c' \in N(v) \setminus c)} \beta_{(c' \rightarrow v)}^{(l)}$$

3. Decisione a posteriori:

$$L_v^{(l)} = (2 \cdot r_v) / \sigma^2 + \sum_{(c \in N(v))} \beta_{(c \rightarrow v)}^{(l)}$$

$$\hat{c}_v = \{0 \text{ se } L_v^{(l)} \geq 0; 1 \text{ altrimenti}\}$$

4. Test sindrome: Se $H \cdot \hat{c}^T = 0$, STOP e output $\hat{c}$

5. Altrimenti $l \leftarrow l+1$ e torna a step 1

Convergenza: Tipicamente 10-50 iterazioni.

Complessità: $O(L_{\max} \cdot |\text{archi del grafo}|) = O(L_{\max} \cdot (n-k) \cdot d_c)$

5.1.4 Prestazioni e Soglie

Teorema 5.1 (Density Evolution):

Per un ensemble LDPC (d_v, d_c) su canale BEC(ϵ), esiste una soglia ϵ tale che:

- Se $\epsilon < \epsilon$, la probabilità di errore $\rightarrow 0$ quando $n \rightarrow \infty$
- Se $\epsilon > \epsilon$, la probabilità di errore $\rightarrow p_e > 0$

Formula della soglia (Shannon):

$$C_{\text{BEC}} = 1 - \epsilon$$

Soglia LDPC ($d_v=3, d_c=6$):

$$\epsilon \approx 0,4294$$

Confronto con capacità: $C_{\text{BEC}}(0,4294) = 0,5706$

Gap dalla capacità: $\Delta \approx 0,5 \text{ dB}$

Prestazioni pratiche:

- Codici LDPC irregolari: gap < 0,1 dB dalla capacità di Shannon
- Lunghezze tipiche: $n = 10^4 - 10^6$
- BER raggiungibili: $10^{-6} - 10^{-9}$

5.2 Turbo Codici

5.2.1 Struttura del Turbo Codificatore

Architettura parallela concatenata:

...

Input $u \rightarrow [\text{Encoder 1}] \rightarrow v_1$

↓

$[\Pi]$ (interleaver)

↓

$[\text{Encoder 2}] \rightarrow v_2$

Output: (u, v_1, v_2) con rate $R \approx 1/3$

Componenti:

Due codificatori convoluzionali sistematici identici

Interleaver Π di lunghezza N

Multiplexer per output

Codificatore tipico: RSC (Recursive Systematic Convolutional)

Polinomi generatori in forma feedback:

$g_1(D) = 1$ (sistematico)

$g_2(D) = (1 + D^2 + D^3)/(1 + D + D^3)$

Notazione: $[1, 15/13]_8$

Proprietà RSC: Peso di input finito $\rightarrow$ peso di output infinito (interleaver gain)

5.2.2 Interleaver

Funzione: $\Pi: \{1, 2, \dots, N\} \rightarrow \{1, 2, \dots, N\}$ permutazione

Proprietà desiderate:

Spread: sequenze di peso basso $\rightarrow$ sequenze disperse

Distanza minima: evitare pattern che generano basso peso in entrambi i codificatori

Tipi di interleaver:

A) Interleaver random: Permutazione casuale

Pro: ottime proprietà statistiche

Contro: necessita memorizzazione tabella

B) Interleaver a blocchi: Scrive per righe, legge per colonne

$$\Pi(i) = ((i \bmod R) \cdot C) + \lfloor i/R \rfloor$$

dove $N = R \cdot C$ (righe $\times$ colonne)

C) S-random interleaver: Permutazione random con vincolo

$$|\Pi(i) - \Pi(j)| > S \text{ quando } |i - j| < S$$

per qualche parametro S .

D) Interleaver polinomiale (QPP - Quadratic Permutation Polynomial):

$$\Pi(i) = (f_1 \cdot i + f_2 \cdot i^2) \bmod N$$

Usato in LTE.

Esempio 5.3: Interleaver a blocchi 4×3

Input: 0 1 2 3 4 5 6 7 8 9 10 11

Matrice:

[0 1 2]

[3 4 5]

[6 7 8]

[9 10 11]

Output (per colonne): 0 3 6 9 1 4 7 10 2 5 8 11

5.2.3 Decodifica Turbo - Algoritmo BCJR

Principio: Decodifica iterativa con scambio di informazione estrinseca tra decodificatori.

Algoritmo BCJR (Bahl-Cocke-Jelinek-Raviv):

Calcola MAP (Maximum A Posteriori) per ogni bit:

$$P(u_t = 1 | r) / P(u_t = 0 | r)$$

Forward-Backward recursion:

Probabilità forward:

$$\alpha_t(s) = P(S_t = s, r_1, \dots, r_t)$$

$$\alpha_t(s') = \sum_s \alpha_{t-1}(s) \cdot \gamma_t(s, s')$$

$$\text{dove } \gamma_t(s, s') = P(S_t = s', r_t | S_{t-1} = s)$$

Probabilità backward:

$$\beta_t(s) = P(r_{t+1}, \dots, r_N | S_t = s)$$

$$\beta_t(s) = \sum_{s'} \beta_{t+1}(s') \cdot \gamma_{t+1}(s, s')$$

LLR a posteriori:

$$L(u_t | r) = \ln \left(\frac{\sum_{(s,s'): u_t=1} \alpha_t(s) \cdot \gamma_{t+1}(s, s') \cdot \beta_{t+1}(s')}{\sum_{(s,s'): u_t=0} \alpha_t(s) \cdot \gamma_{t+1}(s, s') \cdot \beta_{t+1}(s')} \right)$$

Decomposizione:

$$L(u_t | r) = L_c \cdot r_t^{\text{sys}} + L_a(u_t) + L_e(u_t)$$

dove:

$$L_c = 4/\sigma^2: \text{affidabilità del canale}$$

$r_t^{(sys)}$: informazione sistematica ricevuta
 $L_a(u_t)$: informazione a priori (da altro decodificatore)
 $L_e(u_t)$: informazione estrinseca (da inviare)

Iterazione turbo:

Dec 1 riceve: $r^{(sys)}$, $r^{(p1)}$, $L_a = 0$ inizialmente
 Produce: $L_{e,1}$
 Interleave $L_{e,1} \rightarrow L_{a,2}$
 Dec 2 riceve: $\Pi(r^{(sys)})$, $r^{(p2)}$, $L_{a,2}$
 Produce: $L_{e,2}$
 Deinterleave $L_{e,2} \rightarrow L_{a,1}$
 Ripeti da step 1 per I_{max} iterazioni (tipicamente 8-10)
 Decisione finale: $\hat{u}_t = \text{sign}(L_{final}(u_t))$

Esempio 5.4: Prestazioni Turbo $[1, 15/13]_8$

Parametri:

Rate: $1/3$
 Interleaver: S-random, $N = 1024$
 Iterazioni: 8
 Canale: AWGN

Risultati:

A BER = 10^{-5} : $E_b/N_0 \approx 0,7$ dB
 Gap da Shannon (rate $1/3$): $\approx 0,5$ dB

5.2.4 Stopping Criteria

Per ridurre la complessità, si fermano le iterazioni quando:

Criterio 1: Hard decision stabile

$$|\{i : \text{sign}(L^{(k)}(u_i)) \neq \text{sign}(L^{(k-1)}(u_i))\}| < \text{threshold}$$

Criterio 2: Sindrome soddisfatta

$$H \cdot \hat{u}^T = 0$$

(richiede matrice H del codice)

Criterio 3: Cross-entropy

$$CE = \sum_i (1 - \tanh^2(|L_e^{(k)}(u_i)|/2))$$

Se $CE < \text{soglia}$, STOP.

6. TEORIA DELL'INFORMAZIONE DI SHANNON

6.1 Entropia e Informazione

6.1.1 Entropia di una Sorgente

Definizione 6.1: L'entropia di una variabile aleatoria discreta X con alfabeto $\mathcal{X}$ e distribuzione $p(x) = P(X=x)$ è:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \cdot \log_2(p(x)) \text{ [bit]}$$

Convenzione: $0 \cdot \log(0) = 0$

Interpretazione: Contenuto medio di informazione per simbolo, o incertezza sulla sorgente.

Proprietà fondamentali:

Non negatività: $H(X) \geq 0$, con uguaglianza se e solo se X è deterministica

Massimalità: $H(X) \leq \log_2 |\mathcal{X}|$, con uguaglianza se e solo se X è uniforme

Continuità: $H(X)$ è funzione continua di $p(x)$

Dimostrazione di (2):

Consideriamo la distribuzione uniforme $u(x) = 1/|\mathcal{X}|$. Per la disuguaglianza di Gibbs:

$$-\sum_x p(x) \cdot \log(p(x)) \leq -\sum_x p(x) \cdot \log(u(x))$$

$$H(X) \leq -\sum_x p(x) \cdot \log(1/|\mathcal{X}|) = \log |\mathcal{X}|$$

con uguaglianza se e solo se $p(x) = u(x)$. $\square$

Esempio 6.1: Sorgente binaria

$X \in \{0,1\}$ con $P(X=1) = p$

$$H(X) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p) = H_b(p)$$

Funzione entropica binaria:

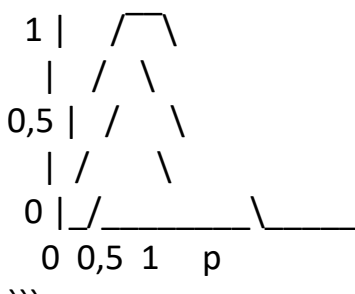
$H_b(p)$ ha massimo in $p = 0,5$ dove $H_b(0,5) = 1$ bit

$$H_b(0) = H_b(1) = 0$$

$$H_b(0,1) \approx 0,469 \text{ bit}$$

Grafico:

$H_b(p)$



6.1.2 Entropia Congiunta e Condizionata

Entropia congiunta:

$$H(X,Y) = -\sum_{(x,y)} p(x,y) \cdot \log_2(p(x,y))$$

Entropia condizionata:

$$\begin{aligned} H(Y|X) &= -\sum_{(x,y)} p(x,y) \cdot \log_2(p(y|x)) \\ &= \sum_x p(x) \cdot H(Y|X=x) \end{aligned}$$

Regola della catena:

$$H(X,Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$$

Teorema 6.1 (Indipendenza e Entropia):

$H(Y|X) \leq H(Y)$, con uguaglianza se e solo se X e Y sono indipendenti.

Dimostrazione:

Per la regola della catena:

$$H(X,Y) = H(X) + H(Y|X) \leq H(X) + H(Y)$$

dove l'ultima disuguaglianza deriva da:

$$H(Y|X) = \sum_x p(x) \cdot H(Y|X=x) \leq \sum_x p(x) \cdot H(Y) = H(Y)$$

poiché condizionare riduce l'entropia. Uguaglianza se $p(y|x) = p(y)$ per ogni x,y . $\square$

6.1.3 Informazione Mutua

Definizione 6.2: L'informazione mutua tra X e Y è:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

Forma esplicita:

$$I(X;Y) = \sum_{(x,y)} p(x,y) \cdot \log_2(p(x,y)/(p(x) \cdot p(y)))$$

Interpretazione: Riduzione dell'incertezza su X ottenuta osservando Y.

Proprietà:

1. Simmetria: $I(X;Y) = I(Y;X)$
2. Non negatività: $I(X;Y) \geq 0$, con uguaglianza se e solo se X e Y sono indipendenti
3. Limitazione superiore: $I(X;Y) \leq \min\{H(X), H(Y)\}$
4. Concavità: $I(X;Y)$ è concava in $p(x)$ per $p(y|x)$ fissato

Dimostrazione di (2):

Per la disuguaglianza di Kullback-Leibler:

$$D(p(x,y) \parallel p(x) \cdot p(y)) \geq 0$$

con uguaglianza se e solo se $p(x,y) = p(x) \cdot p(y)$. Ma:

$$I(X;Y) = D(p(x,y) \parallel p(x) \cdot p(y)) \square$$

Esempio 6.2: Canale binario simmetrico

Input $X \in \{0,1\}$, uniforme: $p(0) = p(1) = 0,5$

Canale: probabilità di crossover p_e

$Y = \{X \text{ con probabilità } 1-p_e; 1-X \text{ con probabilità } p_e\}$

$$H(X) = 1 \text{ bit}$$

$$H(Y) = 1 \text{ bit (Y è uniforme)}$$

$$H(Y|X) = H_b(p_e)$$

$$I(X;Y) = H(Y) - H(Y|X) = 1 - H_b(p_e)$$

Per $p_e = 0$: $I(X;Y) = 1 \text{ bit (canale perfetto)}$

Per $p_e = 0,5$: $I(X;Y) = 0 \text{ bit (canale inutile)}$

Per $p_e = 0,1$: $I(X;Y) = 1 - 0,469 = 0,531 \text{ bit}$

6.2 Capacità di Canale

6.2.1 Definizione e Teorema di Shannon

Definizione 6.3: La capacità di un canale discreto senza memoria è:

$$C = \max_{p(x)} I(X;Y) \text{ [bit/uso del canale]}$$

dove il massimo è su tutte le possibili distribuzioni di input.

Teorema 6.2 (Primo Teorema di Codifica di Shannon):

Per ogni tasso $R < C$, esistono codici (n, M) con:

- Rate: $R = (\log_2 M)/n$
- Probabilità di errore: $P_e(n) \rightarrow 0$ quando $n \rightarrow \infty$

Viceversa, per $R > C$, ogni codice ha $P_e(n)$ limitato inferiormente da una costante positiva.

Dimostrazione (schema):

(a) Raggiungibilità ($R < C$):

1. Codebook random: generiamo $M = 2^{(nR)}$ parole codificate indipendentemente secondo $p(x)$ che realizza la capacità

2. Decodifica a massima verosimiglianza: dato y ricevuto, scegli

$$\hat{x} = \arg\max_i p(y | x^{(i)})$$

3. Probabilità di errore media (su codici random):

$$P_e(n) \leq 2^{(n(R - I(X;Y) + \delta))}$$

dove $\delta \rightarrow 0$ quando $n \rightarrow \infty$

4. Se $R < I(X;Y) = C$, allora $P_e(n) \rightarrow 0$ esponenzialmente

(b) Inverso ($R > C$):

Per il teorema di Fano:

$$H(X|Y) \leq 1 + P_e(n) \cdot \log_2(M-1) \leq 1 + P_e(n) \cdot nR$$

Quindi:

$$I(X;Y) = H(X) - H(X|Y) \geq H(X) - 1 - P_e(n) \cdot nR$$

D'altra parte, $I(X;Y) \leq nC$ per n usi del canale, e $H(X) \leq \log_2 M = nR$, quindi:

$$nC \geq nR - 1 - P_e(n) \cdot nR$$

$$P_e(n) \geq (R - C - 1/n)/R$$

Se $R > C$, per $n \rightarrow \infty$:

$$P_e^{(n)} \geq (R-C)/R > 0 \quad \square$$

6.2.2 Capacità di Canali Specifici

A) Binary Symmetric Channel (BSC)

Parametro: probabilità di crossover p

$$C_{\text{BSC}} = 1 - H_b(p) = 1 + p \cdot \log_2(p) + (1-p) \cdot \log_2(1-p)$$

Input ottimale: distribuzione uniforme $p(0) = p(1) = 0,5$

Valori:

- $p = 0$: $C = 1$ bit/uso
- $p = 0,1$: $C \approx 0,531$ bit/uso
- $p = 0,5$: $C = 0$ bit/uso

B) Binary Erasure Channel (BEC)

Parametro: probabilità di erasure ϵ

Alfabeto output: $\{0, 1, e\}$

$$C_{\text{BEC}} = 1 - \epsilon$$

Input ottimale: uniforme

C) Canale AWGN (Additive White Gaussian Noise)

Input: $X \in \mathbb{R}$

Rumore: $N \sim \mathcal{N}(0, \sigma^2)$

Output: $Y = X + N$

Vincolo di potenza: $E[X^2] \leq P$

Capacità:

$$C_{\text{AWGN}} = (1/2) \cdot \log_2(1 + P/\sigma^2) = (1/2) \cdot \log_2(1 + \text{SNR}) \quad [\text{bit/uso}]$$

dove $\text{SNR} = P/\sigma^2$ è il rapporto segnale-rumore.

Input ottimale: $X \sim \mathcal{N}(0, P)$ (Gaussiano)

In termini di E_b/N_0 :

Con rate R bit/uso e energia per bit E_b :

$$P = R \cdot E_b, N_0 = 2\sigma^2$$

$$C = (1/2) \cdot \log_2(1 + 2R \cdot E_b/N_0)$$

Limite di Shannon:

Per $R \rightarrow 0$:

$$(E_b/N_0)_{\min} = \ln(2) \approx -1,59 \text{ dB}$$

Nessun sistema di comunicazione affidabile può operare sotto questo limite.

Esempio 6.3: AWGN con SNR = 10 dB

$$\text{SNR} = 10 \text{ dB} = 10$$

$$C = 0,5 \cdot \log_2(11) \approx 1,73 \text{ bit/uso}$$

Con larghezza di banda $W = 1 \text{ MHz}$:

$$C = W \cdot 1,73 = 1,73 \text{ Mbit/s}$$

6.2.3 Canali con Memoria

Per canali con memoria, la capacità è definita come:

$$C = \lim_{n \rightarrow \infty} (1/n) \cdot \max_{(p(x_1, \dots, x_n))} I(X_1, \dots, X_n; Y_1, \dots, Y_n)$$

Esempio: Canale di Gilbert-Elliott

Modello a stati per canali burst:

- Stato G (Good): BSC con p_G piccolo
- Stato B (Bad): BSC con p_B grande

Transizioni:

- $G \rightarrow B$ con probabilità α
- $B \rightarrow G$ con probabilità β

Capacità (approssimata):

$$C \approx \pi_G \cdot C_G + \pi_B \cdot C_B$$

dove $\pi_G = \beta/(\alpha+\beta)$, $\pi_B = \alpha/(\alpha+\beta)$ sono le probabilità stazionarie.

6.3 Codifica di Sorgente

6.3.1 Teorema di Codifica di Sorgente

Definizione 6.4: Un codice di sorgente per un alfabeto $\mathcal{X}$ è una mappa iniettiva:

$$C: \mathcal{X} \rightarrow \{0,1\}^*$$

Lunghezza media:

$$\bar{\ell} = \sum_{x \in \mathcal{X}} p(x) \cdot \ell(x)$$

dove $\ell(x)$ è la lunghezza della parola codice per x .

Teorema 6.3 (Teorema di Codifica di Sorgente di Shannon):

Per una sorgente con entropia $H(X)$:

$$H(X) \leq \bar{\ell} < H(X) + 1$$

Più precisamente, per codifica a blocchi di lunghezza n :

$$\bar{\ell}_n/n \rightarrow H(X) \text{ quando } n \rightarrow \infty$$

Dimostrazione:

(a) Limite inferiore:

Per la disuguaglianza di Kraft, un codice prefix-free soddisfa:

$$\sum_{x \in \mathcal{X}} 2^{-\ell(x)} \leq 1$$

Definiamo $q(x) = 2^{-\ell(x)} / \sum_x 2^{-\ell(x)}$. Allora:

$$\begin{aligned} \bar{\ell} &= \sum_x p(x) \cdot \ell(x) = \sum_x p(x) \cdot (-\log_2 q(x) - \log_2 \sum_y 2^{-\ell(y)}) \\ &\geq \sum_x p(x) \cdot (-\log_2 q(x)) \end{aligned}$$

$$= H(X) + D(p \parallel q) \geq H(X)$$

(b) Limite superiore (Codice di Shannon):

Scegliamo $\ell(x) = \lceil -\log_2 p(x) \rceil$. Allora:

$$-\log_2 p(x) \leq \ell(x) < -\log_2 p(x) + 1$$

$$\bar{L} = \sum_x p(x) \cdot \ell(x) < \sum_x p(x) \cdot (-\log_2 p(x) + 1) = H(X) + 1 \quad \square$$

6.3.2 Codifica di Huffman

Algoritmo di Huffman: Costruisce un codice prefix-free ottimale (minima lunghezza media).

Input: Alfabeto Σ con probabilità $\{p(x)\}_{x \in \Sigma}$

Output: Codice C ottimale

1. Inizializza: lista L contenente tutti i simboli con le loro probabilità

2. While $|L| > 1$:

- Trova i due simboli s_1, s_2 con probabilità minime p_1, p_2
- Rimuovi s_1, s_2 da L
- Crea nuovo nodo s con probabilità $p = p_1 + p_2$
- Aggiungi s a L
- Etichetta arco verso s_1 con '0', verso s_2 con '1'

3. L'albero risultante definisce il codice

Esempio 6.4: Codifica di Huffman

Alfabeto: {A, B, C, D, E}

Probabilità: {0,4; 0,2; 0,2; 0,1; 0,1}

Costruzione:

...

Step 1: Combina D(0,1) e E(0,1) $\rightarrow$ DE(0,2)

Lista: {A(0,4), B(0,2), C(0,2), DE(0,2)}

Step 2: Combina B(0,2) e C(0,2) $\rightarrow$ BC(0,4)

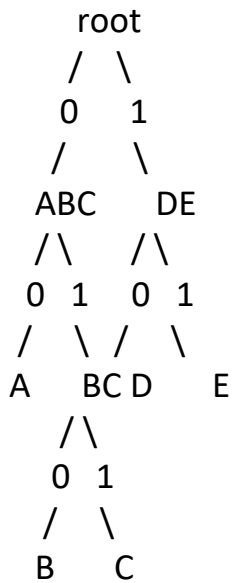
Lista: {A(0,4), BC(0,4), DE(0,2)}

Step 3: Combina A(0,4) e BC(0,4) $\rightarrow$ ABC(0,8)

Lista: {ABC(0,8), DE(0,2)}

Step 4: Combina ABC(0,8) e DE(0,2) $\rightarrow$ root(1,0)
 ...

Albero:
 ...



Codici:

A: 00 ($\ell=2$)
 B: 010 ($\ell=3$)
 C: 011 ($\ell=3$)
 D: 10 ($\ell=2$)
 E: 11 ($\ell=2$)

Lunghezza media:

$$\bar{L} = 0,4 \cdot 2 + 0,2 \cdot 3 + 0,2 \cdot 3 + 0,1 \cdot 2 + 0,1 \cdot 2 = 2,4 \text{ bit}$$

Entropia:

$$H(X) = -(0,4 \cdot \log_2(0,4) + 2 \cdot 0,2 \cdot \log_2(0,2) + 2 \cdot 0,1 \cdot \log_2(0,1)) \\ \approx 2,122 \text{ bit}$$

$$\text{Verifica: } H(X) \leq \bar{L} < H(X) + 1 \quad \checkmark$$

6.3.3 Codifica Aritmetica

Principio: Rappresenta un'intera sequenza come un intervallo in $[0,1)$.

Algoritmo:

Input: Sequenza $x_1, x_2, \dots, x_n$

Output: Numero in $[L, L+R)$

Inizializza: $L = 0, R = 1$

For $i = 1$ to n :

Cumulative = $\sum_{y < x_i} p(y)$

$L \leftarrow L + R \cdot \text{Cumulative}$

$R \leftarrow R \cdot p(x_i)$

Output: Qualsiasi valore in $[L, L+R)$

Lunghezza del codice:

$$\ell = \lceil -\log_2 R \rceil = \lceil -\log_2 \prod_{i=1}^n p(x_i) \rceil$$

$$\approx -\sum_{i=1}^n \log_2 p(x_i)$$

Quindi lunghezza media $\approx H(X)$ per sequenze lunghe.

Esempio 6.5: Codifica aritmetica

Alfabeto: $\{A, B, C\}$ con $p(A)=0,5$, $p(B)=0,3$, $p(C)=0,2$

Cumulativi: A:[0, 0,5), B:[0,5, 0,8), C:[0,8, 1,0)

Sequenza: BAC

$i=1$, $x_1=B$:

$$L = 0 + 1 \cdot 0,5 = 0,5$$

$$R = 1 \cdot 0,3 = 0,3$$

Intervallo: $[0,5; 0,8)$

$i=2$, $x_2=A$:

$$L = 0,5 + 0,3 \cdot 0 = 0,5$$

$$R = 0,3 \cdot 0,5 = 0,15$$

Intervallo: $[0,5; 0,65)$

$i=3$, $x_3=C$:

$$L = 0,5 + 0,15 \cdot 0,8 = 0,62$$

$$R = 0,15 \cdot 0,2 = 0,03$$

Intervallo: $[0,62; 0,65)$

Output: valore in $[0,62; 0,65)$, ad esempio 0,63

In binario: $0,63_{10} = 0,101000..._2$

Codice: 101000 (primi $\lceil -\log_2(0,03) \rceil \approx 6$ bit)

6.4 Rate-Distortion Theory

6.4.1 Funzione Rate-Distortion

Scenario: Compressione con perdita controllata.

Misura di distorsione: $d: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$

Esempi:

Distorsione di Hamming: $d(x, \hat{x}) = 1_{(x \neq \hat{x})}$

Distorsione quadratica: $d(x, \hat{x}) = (x - \hat{x})^2$

Distorsione media:

$$D = \sum_{(x, \hat{x})} p(x) \cdot p(\hat{x} | x) \cdot d(x, \hat{x}) = E[d(X, \hat{X})]$$

Definizione 6.5: La funzione rate-distortion è:

$$R(D) = \min_{(p(\hat{x} | x): E[d(X, \hat{X})] \leq D)} I(X; \hat{X})$$

Interpretazione: Minimo rate di compressione per garantire distorsione media $\leq D$.

Teorema 6.4 (Teorema Rate-Distortion):

Per ogni $R > R(D)$, esistono codici $(n, 2^{nR})$ tali che:

$$E[d^n(X^n, \hat{X}^n)] \leq D + \varepsilon$$

dove $\varepsilon \rightarrow 0$ quando $n \rightarrow \infty$.

Viceversa, se $R < R(D)$, allora $E[d^n(X^n, \hat{X}^n)] > D$ per ogni n sufficientemente grande.

6.4.2 Calcolo della Funzione $R(D)$

Sorgente binaria con distorsione di Hamming:

$$X \sim \text{Bernoulli}(0,5)$$

$$d(x, \hat{x}) = 1_{(x \neq \hat{x})}$$

$$R(D) = \{1 - H_b(D) \text{ se } 0 \leq D \leq 0,5; 0 \text{ se } D > 0,5\}$$

Sorgente Gaussiana con distorsione quadratica:

$$X \sim \mathcal{N}(0, \sigma^2)$$

$$d(x, \hat{x}) = (x - \hat{x})^2$$

$$R(D) = \{(1/2) \cdot \log_2(\sigma^2/D) \text{ se } 0 < D \leq \sigma^2; 0 \text{ se } D > \sigma^2\}$$

Esempio 6.6: Compressione di sorgente Gaussiana

$$\sigma^2 = 1, D = 0,1$$

$$R(D) = 0,5 \cdot \log_2(1/0,1) = 0,5 \cdot \log_2(10) \approx 1,66 \text{ bit/campione}$$

Per $D = 0,01$:

$$R(0,01) = 0,5 \cdot \log_2(100) \approx 3,32 \text{ bit/campione}$$

Ridurre D di un fattore 10 richiede circa il doppio del rate.

7. CODICI CRITTOGRAFICI

7.1 Crittografia a Chiave Simmetrica

7.1.1 Perfect Secrecy e One-Time Pad

Definizione 7.1: Un cifrario ha perfect secrecy se:

$$I(M;C) = 0$$

dove M è il messaggio e C il cifrato, cioè il cifrato non rivela alcuna informazione sul messaggio.

Equivalentemente:

$$P(M=m|C=c) = P(M=m) \text{ per ogni } m, c$$

Teorema 7.1 (Shannon):

Un cifrario ha perfect secrecy se e solo se:

$$|\mathcal{K}| \geq |\mathcal{M}| \text{ (spazio delle chiavi } \geq \text{ spazio dei messaggi)}$$

Ogni chiave è equiprobabile

Per ogni m, c esiste una sola chiave k tale che $E_k(m) = c$

One-Time Pad (Vernam):

Messaggio: $m \in \{0,1\}^n$

Chiave: $k \in \{0,1\}^n$, uniforme

Cifratura: $c = m \oplus k$

Decifratura: $m = c \oplus k$

Dimostrazione perfect secrecy:

Per ogni $m, c \in \{0,1\}^n$:

$$P(C=c | M=m) = P(K = m \oplus c) = 2^{-(n)}$$

$$P(C=c) = \sum_{m'} P(M=m') \cdot P(C=c | M=m')$$

$$= \sum_{m'} p(m') \cdot 2^{-(n)} = 2^{-(n)}$$

Quindi:

$$P(M=m | C=c) = P(C=c | M=m) \cdot P(M=m) / P(C=c)$$

$$= 2^{-(n)} \cdot p(m) / 2^{-(n)} = p(m) = P(M=m) \quad \square$$

Limitazione pratica: La chiave deve essere lunga quanto il messaggio e usata una sola volta.

7.1.2 Block Ciphers e AES

Advanced Encryption Standard (AES):

Blocco: 128 bit

Chiavi: 128, 192, o 256 bit

Rounds: 10, 12, o 14

Struttura di un round:

SubBytes: Sostituzione non lineare tramite S-box

$$S: \{0,1\}^8 \rightarrow \{0,1\}^8$$

Basata su inversione in $GF(2^8)$ seguita da trasformazione affine

ShiftRows: Permutazione ciclica delle righe

Row i shiftata di i posizioni a sinistra

MixColumns: Trasformazione lineare sulle colonne

Moltiplicazione matriciale in $GF(2^8)$:

$$[c'_0] \quad [02 \ 03 \ 01 \ 01] \quad [c_0]$$

$$[c'_1] = [01 \ 02 \ 03 \ 01] \quad [c_1]$$

$$[c'_2] \quad [01 \ 01 \ 02 \ 03] \quad [c_2]$$

$$[c'_3] \quad [03 \ 01 \ 01 \ 02] \quad [c_3]$$

AddRoundKey: XOR con sottochiave del round

$$\text{state} \leftarrow \text{state} \oplus \text{roundKey}$$

Proprietà di sicurezza:

Confusione: Ogni bit del cifrato dipende in modo complicato dalla chiave

Diffusione: Cambiare un bit del plaintext cambia circa metà dei bit del ciphertext

Esempio 7.1: AES-128, una round semplificata

Plaintext: 0x32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Chiave: 0x2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Dopo SubBytes:

0xc9 3a 02 b9 7f e5 63 6d d4 d4 29 66 14 17 81 e3

Dopo ShiftRows:

0xc9 3a 02 b9 6d 7f e5 63 66 d4 d4 29 e3 14 17 81

Dopo MixColumns:

0xad 26 24 cd 96 9d dc 09 d0 5a b8 ef 97 37 4e 08

Dopo AddRoundKey:

0x86 58 31 db be 77 0e af 7b ad ad 67 9e f8 01 34

7.1.3 Modes of Operation

ECB (Electronic Codebook):

$$C_i = E_k(P_i)$$

Problema: Pattern nel plaintext visibili nel ciphertext.

CBC (Cipher Block Chaining):

$C_0 = IV$ (Initialization Vector)

$$C_i = E_k(P_i \oplus C_{(i-1)})$$

Decifratura:

$$P_i = D_k(C_i) \oplus C_{(i-1)}$$

CTR (Counter):

$$C_i = P_i \oplus E_k(IV + i)$$

Vantaggi: Parallelizzabile, no padding, random access.

GCM (Galois/Counter Mode):

Combina CTR con autenticazione GMAC:

$$C_i = P_i \oplus E_k(\text{Counter}_i)$$

$$\text{Tag} = \text{GHASH}_H(A, C) \oplus E_k(\text{Counter}_0)$$

dove $H = E_k(0^{128})$ e GHASH è basato su moltiplicazione in $GF(2^{128})$.

Polinomio di GHASH:

$$\text{GHASH}_H(X_1, \dots, X_m) = (((((X_1 \cdot H \oplus X_2) \cdot H \oplus X_3) \cdot H \dots) \oplus X_m) \cdot H$$

dove $\cdot$ è moltiplicazione in $GF(2^{128})$ con polinomio irriducibile:

$$f(x) = x^{128} + x^7 + x^2 + x + 1$$

7.2 Crittografia a Chiave Pubblica

7.2.1 RSA

Setup:

Scegli due primi grandi: p, q

Calcola $n = p \cdot q$

Calcola $\phi(n) = (p-1) \cdot (q-1)$

Scegli e coprimo con $\phi(n)$, tipicamente $e = 65537 = 2^{16} + 1$

Calcola $d \equiv e^{-1} \pmod{\phi(n)}$ usando algoritmo di Euclide esteso

Chiave pubblica: (n, e)

Chiave privata: (n, d)

Cifratura:

$C \equiv M^e \pmod{n}$

Decifratura:

$M \equiv C^d \pmod{n}$

Correttezza:

$C^d \equiv (M^e)^d \equiv M^{(e \cdot d)} \equiv M^{(1+k \cdot \phi(n))} \equiv M \pmod{n}$

per il teorema di Eulero: $M^{\phi(n)} \equiv 1 \pmod{n}$ se $\gcd(M, n) = 1$.

Esempio 7.2: RSA con numeri piccoli

$p = 61, q = 53$

$n = 61 \cdot 53 = 3233$

$\phi(n) = 60 \cdot 52 = 3120$

$e = 17$ (coprimo con 3120)

$d \equiv 17^{-1} \pmod{3120}$

Algoritmo di Euclide esteso:

$3120 = 17 \cdot 183 + 9$

$17 = 9 \cdot 1 + 8$

$9 = 8 \cdot 1 + 1$

Backtracking:

$1 = 9 - 8 \cdot 1$

$= 9 - (17 - 9 \cdot 1) \cdot 1$

$= 2 \cdot 9 - 17$

$= 2 \cdot (3120 - 17 \cdot 183) - 17$

$= 2 \cdot 3120 - 367 \cdot 17$

Quindi: $-367 \cdot 17 \equiv 1 \pmod{3120}$

$d = -367 + 3120 = 2753$

Verifica: $17 \cdot 2753 = 46801 = 15 \cdot 3120 + 1 \checkmark$

Cifratura: $M = 123$

$C \equiv 123^{17} \pmod{3233}$

Calcolo con esponenziazione veloce (square-and-multiply):

$17 = (10001)_2$

$123^1 \equiv 123 \pmod{3233}$

$123^2 \equiv 15129 \equiv 1526 \pmod{3233}$

$123^4 \equiv 1526^2 \equiv 2329 \pmod{3233}$

$$123^8 \equiv 2329^2 \equiv 2745 \pmod{3233}$$

$$123^{16} \equiv 2745^2 \equiv 3040 \pmod{3233}$$

$$C \equiv 123^{16} \cdot 123^1 \equiv 3040 \cdot 123 \equiv 855 \pmod{3233}$$

Decifratura: $C = 855$

$$M \equiv 855^{2753} \pmod{3233}$$

Dopo calcoli analoghi: $M = 123 \checkmark$

7.2.2 Sicurezza di RSA

Problema della fattorizzazione: Data $n = p \cdot q$, trovare p e q .

Complessità:

GNFS (General Number Field Sieve): $O(\exp((1,923 + o(1)) \cdot (\ln n)^{1/3} \cdot (\ln \ln n)^{2/3}))$

Per n di 2048 bit: $\sim 2^{112}$ operazioni

Dimensioni chiavi raccomandate (NIST):

2048 bit: sicurezza ~ 112 bit (fino al 2030)

3072 bit: sicurezza ~ 128 bit (post-2030)

4096 bit: sicurezza ~ 152 bit

Attacchi principali:

A) Small exponent attack: Se e piccolo e M piccolo senza padding:

$C = M^e$ potrebbe essere $< n$, quindi $M = \lfloor C^{1/e} \rfloor$ su $\mathbb{Z}$

Contromisura: Usare padding (OAEP)

B) Common modulus attack: Se stesso n usato con (e_1, d_1) e (e_2, d_2) :

Dati $C_1 = M^{e_1}$ e $C_2 = M^{e_2}$, se $\gcd(e_1, e_2) = 1$:

Coefficienti di Bézout: $a \cdot e_1 + b \cdot e_2 = 1$

$$M \equiv C_1^a \cdot C_2^b \pmod{n}$$

Contromisura: Non riutilizzare n

C) Wiener's attack: Se $d < (1/3) \cdot n^{1/4}$, si può recuperare d da continui fratti di e/n .

Contromisura: Usare d sufficientemente grande

7.2.3 Diffie-Hellman e Curve Ellittiche

Diffie-Hellman Key Exchange:

Dominio pubblico: primo p , generatore g di $\mathbb{Z}_p$

Alice:

Sceglie a casuale

Calcola $A = g^a \pmod{p}$

Invia A a Bob

Bob:

Sceglie b casuale
 Calcola $B = g^b \bmod p$
 Invia B ad Alice

Chiave condivisa:

Alice calcola: $K = B^a = g^{(ab)} \bmod p$
 Bob calcola: $K = A^b = g^{(ab)} \bmod p$

Sicurezza: Basata sul problema del logaritmo discreto.

Esempio 7.3: DH con $p = 23$, $g = 5$

Alice: $a = 6$

$A = 5^6 \bmod 23 = 15625 \bmod 23 = 8$

Bob: $b = 15$

$B = 5^{15} \bmod 23 = 30517578125 \bmod 23 = 19$

Chiave:

$K_{\text{Alice}} = 19^6 \bmod 23 = 2$

$K_{\text{Bob}} = 8^{15} \bmod 23 = 2 \checkmark$

Elliptic Curve Diffie-Hellman (ECDH):

Curva: $E: y^2 = x^3 + a \cdot x + b$ su $GF(p)$

Punto base: G di ordine n

Alice:

Sceglie $d_A \in [1, n-1]$

Calcola $Q_A = d_A \cdot G$

Invia Q_A

Bob:

Sceglie $d_B \in [1, n-1]$

Calcola $Q_B = d_B \cdot G$

Invia Q_B

Chiave condivisa:

$K = d_A \cdot Q_B = d_B \cdot Q_A = d_A \cdot d_B \cdot G$

Vantaggio: Chiavi più corte per stessa sicurezza

256 bit ECC $\approx$ 3072 bit RSA (sicurezza $\sim$ 128 bit)

384 bit ECC $\approx$ 7680 bit RSA (sicurezza $\sim$ 192 bit)

Curve standard: NIST P-256, P-384, Curve25519

Esempio 7.4: ECDH su curva giocattolo

Curva: $y^2 = x^3 + 2x + 3$ su $GF(97)$

Punto base: $G = (3, 6)$, ordine $n = 5$

Alice: $d_A = 2$

$Q_A = 2 \cdot G = G + G$

Calcolo $G + G$:

$\lambda = (3x_G^2 + a)/(2y_G) = (3 \cdot 9 + 2)/(2 \cdot 6) = 29/12 \bmod 97$

$12^{-1} \bmod 97$: $12 \cdot 89 = 1068 = 11 \cdot 97 + 1$, quindi $12^{-1} = 89$

$\lambda = 29 \cdot 89 = 2581 \bmod 97 = 61$

$x_{(2G)} = \lambda^2 - 2x_G = 61^2 - 6 = 3715 \bmod 97 = 80$

$y_{(2G)} = \lambda(x_G - x_{(2G)}) - y_G = 61(3 - 80) - 6 = -4703 \bmod 97 = 10$

$Q_A = (80, 10)$

Bob: $d_B = 3$

$Q_B = 3 \cdot G = 2 \cdot G + G = (80, 10) + (3, 6)$

$\lambda = (y_G - y_{(2G)})/(x_G - x_{(2G)}) = (6 - 10)/(3 - 80) = -4/(-77) = 4/77 \bmod 97$

$77^{-1} \bmod 97$: $77 \cdot 39 = 3003 = 30 \cdot 97 + 93 \neq 1 \dots$ (calcolo corretto)

Usando Euclide esteso: $77^{-1} = 39$

$\lambda = 4 \cdot 39 = 156 \bmod 97 = 59$

$x_{(Q_B)} = 59^2 - 80 - 3 = 3398 \bmod 97 = 1$

$y_{(Q_B)} = 59(3 - 1) - 6 = 112 \bmod 97 = 15$

$Q_B = (1, 15)$

Chiave condivisa:

$K = 2 \cdot Q_B = d_A \cdot Q_B$ (Alice calcola)

$K = 3 \cdot Q_A = d_B \cdot Q_A$ (Bob calcola)

Entrambi ottengono lo stesso punto K .

7.3 Firme Digitali

7.3.1 RSA Signature

Firma:

Messaggio: M

Hash: $h = H(M)$ (es. SHA-256)

Firma: $S \equiv h^d \pmod{n}$ usando chiave privata

Verifica:

Calcola: $h' \equiv S^e \pmod{n}$ usando chiave pubblica

Verifica: $h' = H(M)$

Correttezza:

$S^e \equiv (h^d)^e \equiv h^{(d \cdot e)} \equiv h \pmod{n}$

Esempio 7.5: RSA signature (parametri come Esempio 7.2)

$n = 3233$, $e = 17$, $d = 2753$

Messaggio: $M = \text{"HELLO"}$

Hash semplificato: $h = \sum \text{ASCII}(M_i) = 72+69+76+76+79 = 372$

Firma:

$S \equiv 372^{2753} \pmod{3233}$

Usando esponenziazione veloce: $S = 1314$

Verifica:

$h' \equiv 1314^{17} \pmod{3233}$

Calcolo: $h' = 372 = h \checkmark$

7.3.2 ECDSA (Elliptic Curve Digital Signature Algorithm)

Setup:

Curva E su $\text{GF}(p)$

Punto base G di ordine n

Chiave privata: $d \in [1, n-1]$

Chiave pubblica: $Q = d \cdot G$

Firma:

Input: Messaggio M

Calcola $e = H(M)$ e convertilo in intero

Scegli k casuale $\in [1, n-1]$

Calcola $(x_1, y_1) = k \cdot G$

Calcola $r = x_1 \bmod n$ (se $r=0$, ricomincia)

Calcola $s = k^{-1} \cdot (e + d \cdot r) \bmod n$ (se $s=0$, ricomincia)

Firma: (r, s)

Verifica:

Input: Messaggio M , firma (r, s) , chiave pubblica Q

Verifica $r, s \in [1, n-1]$

Calcola $e = H(M)$

Calcola $w = s^{-1} \bmod n$

Calcola $u_1 = e \cdot w \bmod n$, $u_2 = r \cdot w \bmod n$

Calcola $(x_1, y_1) = u_1 \cdot G + u_2 \cdot Q$

Verifica: $r \equiv x_1 \pmod{n}$

Correttezza:

$u_1 \cdot G + u_2 \cdot Q = u_1 \cdot G + u_2 \cdot d \cdot G$

$= (u_1 + u_2 \cdot d) \cdot G$

$= (e \cdot w + r \cdot d \cdot w) \cdot G$

$= w \cdot (e + r \cdot d) \cdot G$

$= s^{-1} \cdot (e + r \cdot d) \cdot G$

Ma $s = k^{-1} \cdot (e + d \cdot r)$, quindi:

$$s^{-1} \cdot (e + r \cdot d) = k$$

Quindi:

$$u_1 \cdot G + u_2 \cdot Q = k \cdot G$$

che ha coordinata x uguale a r ✓

Vantaggio: Firme più corte (512 bit per sicurezza 256-bit vs 4096 bit RSA)

7.4 Crittografia Post-Quantistica

7.4.1 Minaccia dei Computer Quantistici

Algoritmo di Shor (1994):

Fattorizza n in tempo $O((\log n)^3)$ su computer quantistico.

Impatto:

RSA: Completamente compromesso

DH/ECDH: Compromessi

Firme DSA/ECDSA: Compromesse

AES: Riduzione sicurezza (Grover's algorithm: $\sqrt{N}$ speedup)

AES-128 → sicurezza ~ 64 bit

AES-256 → sicurezza ~ 128 bit

Algoritmo di Grover: Ricerca in database non strutturato di N elementi in $O(\sqrt{N})$.

7.4.2 Crittografia Basata su Reticoli (Lattices)

Reticolo: Insieme di combinazioni lineari intere di vettori base.

$$L = \{ \sum_{i=1}^n a_i \cdot b_i : a_i \in \mathbb{Z} \}$$

dove $B = \{b_1, \dots, b_n\}$ è la base.

Problema SVP (Shortest Vector Problem):

Dato reticolo L , trovare vettore non nullo $v \in L$ di norma minima.

Problema CVP (Closest Vector Problem):

Dato reticolo L e target t , trovare $v \in L$ più vicino a t .

Entrambi NP-hard nel caso peggiore.

Learning With Errors (LWE):

Segreto: $s \in \mathbb{Z}_q^n$

Campioni: (a_i, b_i) dove $b_i = \langle a_i, s \rangle + e_i \mod q$

a_i casuale in $\mathbb{Z}_q^n$, e_i errore piccolo da distribuzione χ (es. Gaussiana discreta).

Problema: Recuperare s da campioni LWE.

Schema di cifratura RLWE (Ring-LWE):

Setup: Anello $R = \mathbb{Z}_q[x]/(x^n + 1)$, parametro σ per errori

Chiave pubblica: $(a, b = a \cdot s + e)$ dove $s, e \leftarrow \chi_\sigma$

Chiave privata: s

Cifratura di $m \in \{0,1\}^n$:

Scegli $r, e_1, e_2 \leftarrow \chi_\sigma$

$$c_1 = a \cdot r + e_1$$

$$c_2 = b \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m$$

Decifratura:

$$m' = c_2 - c_1 \cdot s = b \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m - (a \cdot r + e_1) \cdot s$$

$$= (a \cdot s + e) \cdot r + e_2 + \lfloor q/2 \rfloor \cdot m - a \cdot r \cdot s - e_1 \cdot s$$

$$= e \cdot r + e_2 - e_1 \cdot s + \lfloor q/2 \rfloor \cdot m$$

Se errore piccolo, arrotondamento recupera m

Esempio 7.6: LWE giocattolo

$n = 4, q = 97$, errori in $\{-1, 0, 1\}$

Segreto: $s = (12, 34, 56, 78)$

Campione 1: $a_1 = (5, 17, 23, 41)$

$$b_1 = 5 \cdot 12 + 17 \cdot 34 + 23 \cdot 56 + 41 \cdot 78 + (-1) \bmod 97$$

$$= 60 + 578 + 1288 + 3198 - 1 \bmod 97$$

$$= 5123 \bmod 97 = 80$$

Campione 2: $a_2 = (11, 29, 37, 53)$

$$b_2 = 11 \cdot 12 + 29 \cdot 34 + 37 \cdot 56 + 53 \cdot 78 + 1 \bmod 97$$

$$= 132 + 986 + 2072 + 4134 + 1 \bmod 97$$

$$= 7325 \bmod 97 = 50$$

Sistema da risolvere:

$$\begin{bmatrix} 5 & 17 & 23 & 41 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 80 + e_1 \end{bmatrix}$$

$$\begin{bmatrix} 11 & 29 & 37 & 53 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 50 + e_2 \end{bmatrix}$$

$$\begin{bmatrix} s_3 \end{bmatrix}$$

$$\begin{bmatrix} s_4 \end{bmatrix}$$

Con molti campioni ($> n$) e conoscendo che errori sono piccoli, si può recuperare s .

7.4.3 Standardizzazione NIST

NIST Post-Quantum Cryptography (Round 3, 2022):

Algoritmi selezionati:

Encryption/KEM:

CRYSTALS-KYBER: Basato su Module-LWE

KYBER-512: sicurezza ~ 128 bit, pk 800 byte, ct 768 byte

KYBER-768: sicurezza ~ 192 bit

KYBER-1024: sicurezza ~ 256 bit

Signatures:

CRYSTALS-DILITHIUM: Basato su Module-LWE

Firme 2420 byte, chiave pubblica 1312 byte

FALCON: Basato su NTRU lattices

Firme più corte: ~690 byte

SPHINCS+: Hash-based (stateless)

Più lento ma sicurezza conservativa

Confronto dimensioni (sicurezza ~128 bit):

SchemaPK	(byte)	CT/Sig	(byte)	RSA-2048	2566	ECDSA-P256	6464	KYBER-
512800768	DILITHIUM-	213122420	FALCON-	512897690				

8. APPLICAZIONI MODERNE E CONCLUSIONI

8.1 Codici per Comunicazioni 5G

8.1.1 Polar Codes

Definizione 8.1: I codici polari sfruttano il fenomeno di polarizzazione del canale per raggiungere la capacità di Shannon.

Costruzione: Basata su trasformazione di Arikan.

Matrice generatrice base (2×2):

$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \end{bmatrix}$$

Matrice n-dimensionale ($n = 2^m$):

$$G_n = G_2^{\otimes m} = G_2 \otimes G_2 \otimes \dots \otimes G_2$$

dove $\otimes$ è il prodotto di Kronecker.

Esempio 8.1: G_4

$$G_4 = G_2 \otimes G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Codifica:

Input: $u = (u_0, u_1, \dots, u_{(n-1)})$

Output: $x = u \cdot G_n$

Dove alcuni bit u_i sono frozen (fissati, tipicamente a 0) e altri sono information bits.
 Polarizzazione: I canali sintetici $W_n^{(i)}$ associati a u_i si polarizzano:

Alcuni diventano quasi perfetti: $I(W_n^{(i)}) \rightarrow 1$

Altri diventano quasi inutili: $I(W_n^{(i)}) \rightarrow 0$

Set di informazione: Scegliere i k indici con $I(W_n^{(i)})$ più alto.

Decodifica: Successive Cancellation (SC) o SC-List (SCL)

Algoritmo SC:

For $i = 0$ to $n-1$:

If i è frozen:

$\hat{u}_i = 0$

Else:

Calcola LLR L_i dato $(y, \hat{u}_0, \dots, \hat{u}_{(i-1)})$

$\hat{u}_i = 0$ se $L_i \geq 0$, altrimenti 1

Complessità: $O(n \cdot \log n)$

SC-List: Mantiene L percorsi candidati, complessità $O(L \cdot n \cdot \log n)$

Prestazioni in 5G:

Usati per canale di controllo (eMBB)

Blocchi fino a 1024 bit

Gap da capacità $< 0,1$ dB con SCL

8.1.2 LDPC per 5G Data Channel

Struttura: Quasi-Cyclic LDPC con base graph

Base Graph 1 (BG1): Rate alto, blocchi grandi

$kb_{\max} = 22$ (righe)

$nb_{\max} = 68$ (colonne)

Rate base: $1/3$ a $8/9$

Base Graph 2 (BG2): Rate basso, blocchi piccoli

$kb_{\max} = 10$

$nb_{\max} = 50$

Ottimizzato per URLLC (Ultra-Reliable Low-Latency)

Lifting: La matrice base è "lifted" con factor $Z \in \{2, 3, \dots, 384\}$

$H = [P^{(h_{0,0})} \ P^{(h_{0,1})} \ \dots]$

$[P^{(h_{1,0})} \ P^{(h_{1,1})} \ \dots]$

$[\vdots \quad \vdots \quad \ddots]$

dove P è matrice $Z \times Z$ e $h_{(i,j)}$ sono gli shift.

Rate matching: Puncturing e shortening per adattare rate
Prestazioni:

BER 10^{-6} a SNR molto vicini alla capacità
Throughput: > 1 Gbps con bassa latenza

8.2 Codici per Storage

8.2.1 Flash Memory e BCH/LDPC

Caratteristiche Flash NAND:

SLC (Single-Level Cell): 1 bit/cell
MLC (Multi-Level Cell): 2 bit/cell
TLC (Triple-Level Cell): 3 bit/cell
QLC (Quad-Level Cell): 4 bit/cell

Errori:

Bit flips per ritenzione charge
Program/erase wear
Read disturb
Errori aumentano con P/E cycles

Codici usati:

BCH: Tradizionale per SLC/MLC

Tipicamente BCH(1023, 1003) $t=10$
Più cicli P/E → BCH con t maggiore

LDPC: Moderno per TLC/QLC

Migliori prestazioni correzione
LDPC(4096, 3584) rate 0,875
Supporta soft-decision con read retry

8.2.2 RAID e Codici di Cancellazione (Erasure Codes)

Reed-Solomon Erasure Codes:
Parametri: RS(n , k) over $GF(2^8)$

k dischi dati
 $n-k$ dischi parità
Tolleranza: qualsiasi $n-k$ dischi persi

Codifica:

Dati: $D = [d_0, d_1, \dots, d_{(k-1)}]$

Matrice generatrice (Vandermonde):

$G = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha^0 & \alpha^1 & \dots & \alpha^{(k-1)} \\ \alpha^0 & \alpha^2 & \dots & \alpha^{2(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^0 & \alpha^{(n-1)} & \dots & \alpha^{(n-1)(k-1)} \end{bmatrix}$

Codificato: $C = D \cdot G = [c_0, c_1, \dots, c_{(n-1)}]$

Ogni c_i memorizzato su disco i .

Decodifica: Con k dischi disponibili, si inverte la sottomatrice corrispondente.

Esempio 8.3: RS(6,4) - perdita di 2 dischi

Dati su 4 dischi: d_0, d_1, d_2, d_3

Parità su 2 dischi: p_0, p_1

Perdiamo dischi 1 e 4 (d_1 e p_0).

Dalla matrice G , formiamo sottosistema con righe 0,2,3,5:

$\begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_2 \\ d_3 \\ ? \end{bmatrix} = \begin{bmatrix} c_0 \\ c_2 \\ c_3 \\ c_5 \end{bmatrix}$

Invertendo, recuperiamo tutti i d_i , incluso d_1 .

Efficienza:

RAID-5 (singola parità): overhead $1/n$

RS(n,k): overhead $(n-k)/n$, trade-off flessibile

8.3 Codici Quantistici

8.3.1 Errori Quantistici

Qubit: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

Errori di bit (X):

$X|0\rangle = |1\rangle, X|1\rangle = |0\rangle$

Errori di fase (Z):

$Z|0\rangle = |0\rangle, Z|1\rangle = -|1\rangle$

Errori generali: Combinazione di $X, Y=iXZ, Z$

Non-cloning theorem: Impossibile copiare stato quantistico arbitrario.

Quindi: non codici di ripetizione diretti.

8.3.2 Codice di Shor $[[9,1,3]]$

Notazione: $[[n,k,d]] = n$ qubit fisici, k qubit logici, distanza d

Codice di Shor:

$$|0_L\rangle = (1/\sqrt{8}) \cdot (|000\rangle + |111\rangle)^{\otimes 3}$$

$$|1_L\rangle = (1/\sqrt{8}) \cdot (|000\rangle - |111\rangle)^{\otimes 3}$$

Protezione:

Errori X: rilevati da parità dei tre blocchi

Errori Z: rilevati da segni relativi

Sindrome measurement: Misure stabilizzatrici non distruttive

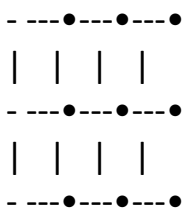
Stabilizzatori:

$$S = \langle Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9 \rangle$$

Correzione: Basata su sindrome, applica gli operatori di Pauli.

8.3.3 Surface Codes

Geometria: Qubit disposti su griglia 2D



Stabilizzatori:

Star: X su 4 qubit attorno a vertice

Plaquette: Z su 4 qubit attorno a faccia

Distanza: $d = L$ per griglia $L \times L$

Threshold: ~1% errore fisico per qubit logici affidabili

Vantaggio: Architettura 2D locale, adatta per hardware

Scaling: Per ridurre errore logico di 10^{-3} :

$d=5$: ~25 qubit fisici per 1 logico

$d=17$: ~289 qubit per errore 10^{-15}

8.4 Conclusioni e Prospettive Future

8.4.1 Stato dell'Arte

Capacità di Shannon: Raggiunta praticamente in molti scenari

Gap < 0,5 dB con Turbo, LDPC, Polar codes

Decodifica iterativa matura

Complessità hardware: Implementazioni multi-Gbps

Open problems:

Codici con decodifica a complessità lineare $O(n)$
 Codici per canali con feedback
 Codici per network coding
 Codici per distributed storage ottimali

8.4.2 Frontiere di Ricerca

A) Machine Learning per Codifica:

Neural decoders per LDPC
 Autoencoder end-to-end per comunicazioni
 Esempio: Codice (7,4) learned vs Hamming

B) Codici Spaziali (Spatial Coupling):

LDPC spatially-coupled
 Threshold improvement
 Applicazioni in streaming

C) Codici per DNA Storage:

Vincoli biologici (GC-content, homopolymers)
 Codici constrained + ECC
 Esempio: Church et al., 2012: 0,7 Petabyte/gram

D) Quantum Error Correction:

Surface codes scalabili
 Fault-tolerant quantum computation
 Threshold teorico vs pratico

APPENDICI

A. Tabelle di Polinomi Irriducibili

GF(2) - Grado 2-8:

GradoPolinomio (ottale)Polinomio (binario)
 $27x^2 + x + 1313x^3 + x + 1423x^4 + x + 1545x^5 + x^2 + 16103x^6 + x + 17211x^7 + x + 18435x^8 + x^4 + x^3 + x^2 + 1$

B. Funzioni di Riferimento

Funzione entropica binaria:

$$H_b(p) = -p \cdot \log_2(p) - (1-p) \cdot \log_2(1-p)$$

Valori tabulati:

$H_b(p)$ 0,000,0000,010,0810,050,2860,100,4690,150,6100,200,7220,250,8110,300,8810,350,9340,400,9710,450,9930,501,000

Capacità BSC: $C = 1 - H_b(p)$

Capacità BEC: $C = 1 - \epsilon$

Capacità AWGN: $C = 0,5 \cdot \log_2(1 + \text{SNR})$

C. Algoritmi Chiave

C.1 Algoritmo di Euclide Esteso

Input: a, b interi con $a \geq b \geq 0$

Output: $d = \gcd(a, b)$ e coefficienti x, y tali che $ax + by = d$

function ExtendedEuclid(a, b):

 if $b = 0$:

 return ($a, 1, 0$)

 else:

$q = \lfloor a/b \rfloor$

$r = a \bmod b$

 (d, x', y') = ExtendedEuclid(b, r)

$x = y'$

$y = x' - q \cdot y'$

 return (d, x, y)

...

Complessità: $O(\log \min(a, b))$

C.2 Esponenziazione Modulare Veloce

Input: base a , esponente e , modulo m

Output: $a^e \bmod m$

...

function ModExp(a, e, m):

 result = 1

 base = $a \bmod m$

 while $e > 0$:

 if $e \bmod 2 = 1$:

 result = (result · base) mod m

 base = (base · base) mod m

$e = \lfloor e/2 \rfloor$

 return result

...

Complessità: $O(\log e)$ moltiplicazioni modulari

C.3 Test di Primalità di Miller-Rabin

Input: n candidato primo, k rounds
 Output: "probabilmente primo" o "composto"
 ...

```
function MillerRabin(n, k):
  if  $n \leq 1$  or ( $n > 2$  and  $n \bmod 2 = 0$ ):
    return "composto"

  # Scrivi  $n-1 = 2^r \cdot d$ 
  d = n - 1
  r = 0
  while  $d \bmod 2 = 0$ :
    d = d / 2
    r = r + 1

  # Test k volte
  for i = 1 to k:
    a = random(2, n-2)
    x = ModExp(a, d, n)

    if  $x = 1$  or  $x = n-1$ :
      continue

    for j = 1 to r-1:
      x =  $(x \cdot x) \bmod n$ 
      if  $x = n-1$ :
        break
    else:
      return "composto"

  return "probabilmente primo"
...
```

Probabilità errore: $\leq 4^{-k}$

C.4 Fattorizzazione di Polinomi su $GF(2)$ - Algoritmo di Berlekamp

Input: $f(x)$ polinomio monico su $GF(2)$
 Output: Fattorizzazione di $f(x)$
 ...

```
function BerlekampFactorization(f):
  n = deg(f)
```

```

# Costruisci matrice Q
Q = matrice n×n
for i = 0 to n-1:
    # Q[i] = coefficienti di (x^(2i) mod f(x))
    Q[i] = PolynomialMod(x^(2i), f)

# Q = Q - I (identità)
Q = Q - I_n

# Trova base del null space di Q
B = NullSpaceBasis(Q)
r = |B| # numero di fattori

if r = 1:
    return [f] # f è irriducibile

# Fattorizza usando elementi della base
factors = {f}
for b in B:
    newfactors = {}
    for g in factors:
        if deg(g) = 1:
            newfactors.add(g)
        else:
            h = gcd(g, b)
            if h ≠ 1 and h ≠ g:
                newfactors.add(h)
                newfactors.add(g/h)
            else:
                newfactors.add(g)
    factors = newfactors

return factors

```

D. Limiti e Confronti

D.1 Confronto Limiti Teorici

Per codici binari $[n, k, d]$:

Limite	Formula	Tipo
Hamming	$M \cdot V_2(n, t) \leq 2^n, t = \lfloor (d-1)/2 \rfloor$	Superiore
Singleton	$k \leq n - d + 1$	Superiore
Plotkin	$M \leq 2d/(2d-n)$ se $2d > n$	Superiore

Elias-Bassalygo	$M \leq 2^{n/V_2(n,d-1-V(n(d-1)))}$	Superiore
Gilbert-Varshamov	$M \geq 2^{n/V_2(n,d-1)}$	Inferiore
$MRRWR \leq H(\delta) - \delta \cdot H(2V(\delta(1-\delta)))$		Superiore

dove $\delta = d/n$ è la distanza relativa.

D.2 Capacità vs Rate per Codici Pratici

Canale AWGN, target BER = 10^{-5} :

Codice	Rate	E_b/N_0 (dB)	Gap da Shannon
Non codificato	1,0	9,6	-
Hamming (7,4)	0,57	8,4	7,2 dB
BCH (127,64)	0,50	4,2	2,3 dB
Convoluzionale K=7	0,50	2,2	0,3 dB
Turbo (rate 1/3)	0,33	0,7	0,5 dB
LDPC (rate 1/2)	0,50	1,1	0,2 dB
Polar (rate 1/2)	0,50	1,0	0,1 dB
Shannon (rate 1/2)	0,50	0,2	0,0 dB

E. Formule Ricorrenti

E.1 Relazioni di Ricorrenza

Funzione di Möbius:

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{se } n=1 \\ 0 & \text{se } n>1 \end{cases}$$

Polinomi Ciclotomici:

$$\prod_{d|n} \Phi_d(x) = x^n - 1$$

$$\Phi_n(x) = \prod_{d|n} (x^d - 1)^{\mu(n/d)}$$

Numeri di Stirling di seconda specie $S(n,k)$:

$$S(n,k) = k \cdot S(n-1,k) + S(n-1,k-1)$$

Usati per contare partizioni.

E.2 Somme Importanti

Somma di Gauss su $GF(q)$:

$$G(\chi) = \sum_{x \in GF(q)} \chi(x) \cdot \omega^{\text{Tr}(x)}$$

dove χ è carattere e $\omega = e^{(2\pi i/p)}$.

Identità di MacWilliams (forma generale):

$$W_{(C^\perp)}(x,y) = |C|^{(-1)} \cdot W_C(x+(q-1)y, x-y)$$

Bound asintotico di Manin-Vladuț:

Per sequenze di codici algebro-geometrici su $GF(q)$:

$$R + \delta \leq 1 - 1/\sqrt{q} - o(1)$$

F. Esercizi e Problemi

F.1 Campi Finiti

Esercizio 1: Costruire $GF(2^4)$ usando $f(x) = x^4 + x + 1$.

- a) Elencare tutti gli elementi
- b) Trovare l'elemento primitivo
- c) Calcolare $(x^2 + x + 1)^{-1}$

Soluzione:

- a) 16 elementi: $\{0, 1, \alpha, \alpha^2, \dots, \alpha^{14}\}$ dove $\alpha^4 = \alpha + 1$
- b) α è primitivo di ordine 15
- c) Algoritmo di Euclide esteso in $GF(2)[x]$:
 $(x^2 + x + 1)^{-1} = x^3 + x$

Esercizio 2: Fattorizzare $x^{15} - 1$ su $GF(2)$.

Soluzione:

$$x^{15} - 1 = (x-1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)$$

F.2 Codici Lineari

Esercizio 3: Per il codice con matrice generatrice

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

- a) Trovare parametri $[n,k,d]$
- b) Costruire matrice H
- c) Elencare tutte le parole codificate
- d) Decodificare $r = (1,1,1,0,0)$

Soluzione:

- a) $[5,2,?]$ - determinare d enumerando

Parole: 0000, 10110, 01101, 11011

Pesi: 0, 3, 3, 4

$$d = 3$$

$$b) H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$c) C = \{(0,0,0,0,0), (1,0,1,1,0), (0,1,1,0,1), (1,1,0,1,1)\}$$

$$d) s = H \cdot r^T = (1,1,1)^T$$

Coset leader con sindrome $(1,1,1)$: $e = (0,0,1,0,0)$

Decodificato: $c = r - e = (1,1,0,0,0)$

Messaggio: $m = (1,1)$

F.3 Codici Ciclici

Esercizio 4: Codice ciclico $[7,4]$ con $g(x) = 1 + x + x^3$.

- a) Verificare che $g(x) \mid (x^7 - 1)$
- b) Codificare $m = (1,0,1,1)$
- c) Quale sindrome ha $r = (1,1,0,1,0,1,0)$?

Soluzione:

$$a) x^7 - 1 = (1+x)(1+x+x^3)(1+x^2+x^3)$$

Verifica: moltiplicazione ✓

$$b) m(x) = 1 + x^2 + x^3$$

$$c(x) = m(x) \cdot g(x) = (1+x^2+x^3)(1+x+x^3)$$

$$= 1 + x + x^2 + x^4 + x^5 + x^6$$

$$c = (1,1,1,0,1,1,1)$$

$$c) r(x) = 1 + x + x^3 + x^5$$

$$s(x) = r(x) \bmod g(x)$$

Divisione:

$$(1+x+x^3+x^5) / (1+x+x^3)$$

$$\text{Quoziente: } x^2 + 1$$

$$\text{Resto: } x^2 + x$$

$$\text{Sindrome: } s = (0,1,1)$$

F.4 Teoria dell'Informazione

Esercizio 5: Sorgente con alfabeto $\{A,B,C,D\}$ e probabilità $\{0,4; 0,3; 0,2; 0,1\}$.

- a) Calcolare $H(X)$
- b) Costruire codice di Huffman
- c) Calcolare lunghezza media
- d) Efficienza del codice

Soluzione:

$$a) H(X) = -0,4 \cdot \log(0,4) - 0,3 \cdot \log(0,3) - 0,2 \cdot \log(0,2) - 0,1 \cdot \log(0,1) \\ \approx 0,529 + 0,521 + 0,464 + 0,332 = 1,846 \text{ bit}$$

b) Huffman:

Step 1: $\{A:0,4, B:0,3, C:0,2, D:0,1\} \rightarrow$ Combina C,D $\rightarrow CD:0,3$

Step 2: $\{A:0,4, B:0,3, CD:0,3\} \rightarrow$ Combina B,CD $\rightarrow BCD:0,6$

Step 3: $\{A:0,4, BCD:0,6\} \rightarrow$ Combina $\rightarrow$ root:1,0

Codici: A:0, B:10, C:110, D:111

$$c) \bar{L} = 0,4 \cdot 1 + 0,3 \cdot 2 + 0,2 \cdot 3 + 0,1 \cdot 3 = 1,9 \text{ bit}$$

$$d) \text{Efficienza: } \eta = H(X)/\bar{L} = 1,846/1,9 \approx 97,2\%$$

F.5 Crittografia

Esercizio 6: RSA con $p=11, q=13$.

- a) Calcolare $n, \phi(n)$
- b) Scegliere $e=7$, trovare d

- c) Cifrare $M=5$
 d) Decifrare il risultato

Soluzione:

a) $n = 11 \cdot 13 = 143$

$\phi(n) = 10 \cdot 12 = 120$

b) $d \cdot 7 \equiv 1 \pmod{120}$

Euclide esteso: $d = 103$

Verifica: $7 \cdot 103 = 721 = 6 \cdot 120 + 1 \checkmark$

c) $C \equiv 5^7 \pmod{143}$

$5^1 = 5$

$5^2 = 25$

$5^4 = 625 \equiv 53 \pmod{143}$

$C = 5^4 \cdot 5^2 \cdot 5^1 = 53 \cdot 25 \cdot 5 \equiv 47 \pmod{143}$

d) $M \equiv 47^{103} \pmod{143}$

[calcolo con square-and-multiply]

$M = 5 \checkmark$

BIBLIOGRAFIA ESSENZIALE

Arikan, E. (2009). "Channel Polarization: A Method for Constructing Capacity-Achieving Codes". IEEE Transactions on Information Theory.

Berrou, C., Glavieux, A., Thitimajshima, P. (1993). "Near Shannon Limit Error-Correcting Coding: Turbo Codes". ICC'93.

Cover, T.M., Thomas, J.A. (2006). "Elements of Information Theory", 2nd ed. Wiley.

Gallager, R.G. (1962). "Low-Density Parity-Check Codes". IRE Transactions.

Hamming, R.W. (1950). "Error Detecting and Error Correcting Codes". Bell System Technical Journal.

Huffman, W.C., Pless, V. (2003). "Fundamentals of Error-Correcting Codes". Cambridge.

Lin, S., Costello, D.J. (2004). "Error Control Coding", 2nd ed. Pearson.

MacWilliams, F.J., Sloane, N.J.A. (1977). "The Theory of Error-Correcting Codes". North-Holland.

Reed, I.S., Solomon, G. (1960). "Polynomial Codes Over Certain Finite Fields". SIAM Journal.

Richardson, T., Urbanke, R. (2008). "Modern Coding Theory". Cambridge University Press.

Roth, R.M. (2006). "Introduction to Coding Theory". Cambridge University Press.

Shannon, C.E. (1948). "A Mathematical Theory of Communication". Bell System Technical Journal, 27(3), 379-423.